UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN.

# "SISTEMA DE ESTIMACIÓN Y CONTROL DE PROYECTOS."

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO MECÁNICO ELECTRICISTA
P R E S E N T A :

**FELIPE DE JESÚS PÉREZ ACOSTA**

**ASESOR: ING. ARMANDO AGUILAR MÁRQUEZ**

**CUAUTITLÁN IZCALLI, EDO. DE MEX.** **1997.**

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

# FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES-CUAUTITLAN

DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

DR. JAIME KELLER TORRES
DIRECTOR DE LA FES-CUAUTITLAN
P R E S E N T E .

AT'N: Ing. Rafael Rodriguez Ceballos
Jefe del Departamento de Exámenes
Profesionales de la F.E.S. - C.

Con base en el art. 28 del Reglamento General de Examenes, nos permitimos comunicar a usted que revisamos la TESIS:
"Sistema de estimación y control de proyectos".

que presenta el pasante: Felipe de Jesús Pérez Acosta
con número de cuenta: 7807035-8 para obtener el TITULO de:
Ingeniero Mecánico Electricista.

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E .
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Edo. de Méx., a 10 de diciembre de 1996

| | |
|---|---|
| PRESIDENTE | Ing. Juan Rafael Garibay Bermúdez |
| VOCAL | Ing. José Juan Contreras Espinosa |
| SECRETARIO | Ing. Armando Aguilar Márquez |
| PRIMER SUPLENTE | Ing. José Luz Hernández Castillo |
| SEGUNDO SUPLENTE | Ing. Jorge Altamira Ibarra |

UAE/DEP/VAP/01

# AGRADECIMIENTOS.

## Papá , Mamá:

que DIOS los llene de gracia por su amor, cariño y ejemplo. Su apoyo, consejo y guía, siempre han logrado el soporte, la idea y el camino, para salir adelante.

### A mis hermanos:

| | |
|---|---|
| María Guadalupe, | hermana mamá |
| Adolfo, | hermano mayor. |
| Lorenzo, | hermano amigo |
| Mirsa Gema, | hermana ingeniera |
| Luis Rogelio, | hermano carnal |
| Ana María Caridad, | hermana artista |
| Margarita, | carnala. |
| Diana , | hermanita menor. |
| Francisco Eduardo, | Panchito. |

Mis amados hermanos, gracias por todo.

### A todos mis sobrinos.

### Ing. Juan José Zamudio Vázquez

Agradezco en todo lo que vale su tiempo, consejos y atenciones. Tomar el reto y lograr la meta, fue gracias a usted.

### Ing. Armando Aguilar Márquez.

Mil gracias por su asesoría y por creer en mi sin conocerme.

### A mi *alma mater*, querida U.N.A.M.

### A Mayola Concepción Niño Méndez

Amada esposa gracias por tu apoyo y comprensión y por todo el tiempo que no estuve contigo y con nuestra hija, por elaborar este trabajo, que dedico de nosotros y con mucho cariño a Carolina
Carito hija, lenguetazo y beso de papá.

### En recuerdo vivo de mis abuelitos.

Adolfo Pérez Lerma y Amalia Martínez Estevez
Lorenzo Acosta Macías y Ramona Nito Torres
Don Adolfo y Don Lorenzo a los que conocí por pláticas de mis padres y abuelitas.
Mamamonche donde estés, te quiero mucho y ya; Agüe, a ti también te quiero mucho, les mando un beso y mi cariño a los cuatro.

### A Don Elías Amado Garavito

Padrinazo, que DIOS premie su cariño y atenciones

### Héctor Figueroa Blando

Amigo, como siempre mi amistad sincera y fraterna.

Agradecimiento especial.

Debido a mi mala memoria, les pido una disculpa a todos aquellos, cuyo nombre no aparece, y que dedicaron tiempo, recursos y esfuerzos en esta humilde servidor.

# Introducción.

## Prologo.

Las organizaciones de Investigación de Operaciones (IO), aparecen en las empresas como áreas estratégicas, después de los resultados satisfactorios que estas técnicas generaron, al solucionar gran parte de los problemas de los Comandantes Aliados en la Segunda Guerra Mundial.

A más de cincuenta años de sus inicios, las herramientas estadísticas aplicadas a la industria y los negocios junto con otras técnicas de IO, han dejado hermosos ejemplos de lo que se puede lograr con la optimización de los recursos para abatir costos y buscar la mejora de los productos y servicios. "Hacer más y mejor con menos".

Logrando:

- Ahorro de desperdicios en materiales y esfuerzos;
- Revisión permanente en los procesos para su optimización;
- Desarrollo de efectividad en los grupos de trabajo;
- Innovación en los diseños y modelos.

De estas mejoras la relación gerencia-fuerza de trabajo se ve vitalizada por la participación administrativa y la reedificación de la dignidad y respeto del trabajador. Esta poderosa fuerza motivadora despierta el deseo de cambio y el hambre de capacitación:

- Mejora la comunicación vertical y horizontalmente;
- Mejora de las relaciones jefe-trabajador;
- Promueve el desarrollo personal y el liderazgo;
- Genera la humildad para aprender de los demás.

Asi el potencial creativo e innovador que tiene la fuerza compartida de trabajo crea resultados de satisfacción y reconocimiento :

" Si te ayudo a resolver tus necesidades, yo gano también "

Y es que para toda actividad humana " siempre hay una mejor forma de realizarla", y "todo es susceptible de cambio".

Además hay que considerar que cualquier tipo de actividad humana está sujeta al tiempo y al espacio en que le toca desarrollarse y esto hace que los factores externos siempre cambien, por lo que es importante tener una actitud de adecuación. Los audaces son los que producen cambio, es decir son dueños del futuro por diseño, "planean el futuro".

## Objetivos.

### Objetivo principal:

Desarrollar un sistema de información para la estimación y control de proyectos.

### Objetivos secundarios:

I. Inculcar los conceptos para administrar un proyecto.

II. Crear capacidades sistemáticas en la resolución de problemas.

III. Desarrollar una actitud de prevención de problemas.

IV. Dimensionar un proyecto.

V. Detectar puntos críticos del proyecto en forma oportuna.

VI. Solucionar problemas después de iniciado el proyecto.

VII. Medir la productividad de los recursos humanos.

VIII. Proporcionar información óptima, para la toma de decisiones.

## Justificación.

En una país como el nuestro lleno de retos , es de vital importancia la optimización de los recursos. La tarea se simplifica utilizando una estrategia, que no es otra cosa, que una metodologia aplicada sobre las actividades cotidianas, generalmente menospreciadas, pero que son la estructura fundamental de toda organización. Es imperante la difusión de estas metodologias o sistemas, no sólo a niveles directivos, sino a todos aquellos grupos importantes, como profesionistas, estudiantes y prácticos.

Para solucionar los problemas de estimación , valuación y control de proyectos, lo más importante es la comprensión y aplicación práctica de métodos y procedimientos, creando conciencia del valor de la previsión y el orden.

La computadora es una herramienta, un instrumento del cual podemos aprovechar sus virtudes y caracteristicas; pero todavía no piensa.

## Narrativa por capítulos.

La estructura a seguir en este texto llevará como orden el ciclo de vida de los proyectos y el ciclo de vida del desarrollo de los sistemas de información, así en el capítulo uno hablaremos de algunos conceptos sobre los métodos tradicionales para el diseño, estimación, control y estadísticas de proyectos, después veremos el ciclo de vida clásico del software y poder explicar las técnicas de Cuarta Generación y para finalizar el capítulo explicaremos rápidamente los conceptos que involucra la programación orientada a objetos y a eventos, así como sus principales elementos.

El segundo capítulo nos introducirá a lo que será el método utilizado, ya que veremos qué es un proyecto y qué lo compone, además conoceremos qué es y por qué es necesario y conveniente realizar el sistema.

En el capítulo tres expondremos el análisis del sistema de información el cual se realizará una vez conocido el objetivo del sistema. El análisis incluirá análisis de requerimientos, análisis estructurado y modelización de datos, además surgirán algunos productos intemedios como son: la relación de requerimientos generales, los diagramas de flujos de datos, así como su diccionario de requisitos, diagramas de entidad relación, conceptos sobre diseño lógico de base de datos, diseño físico de base de datos.

El capítulo cuatro enfocará el Diseño del Sistema, ya que este se realiza una vez concluido el análisis, el diseño incluye: el diseño de datos y el diseño estructural, algunos productos intermedios de esta fase son el diccionario de datos, la relación de estructuras de las bases de datos, las especificaciones de la interface con el usuario y el prototipo del software.

Seguido viene el Desarrollo del Sistema, que se divide en la programación y la creación de la base de datos. Veremos la evolución de la programación y los lenguajes computacionales, marcando las características de los que se consideran de cuarta

generación y sus bondades para la creación de software. En la parte de base de datos analizaremos la teoría de bases de datos de manera general, sus elementos, el sistema de manejo de datos, la arquitectura de un sistema relacional y la diferencia para con otras arquitecturas. De esta manera se compondra el capitulo quinto.

Para el sexto capítulo se realizarán las pruebas y la puesta en marcha del sistema, se explicarán las pruebas llamadas de caja blanca y de caja negra. Para la puesta en marcha se especifican segun el caso la forma de capacitar colaboradores o usuarios del sistema y la carga de información para el uso del mismo.

La parte terminal del trabajo es la de las conclusiones y valoración de resultados.

Índice.

**Contenido** **página**


**Agradecimientos** 1

**Introducción** 2

**Prologo** 2

**Objetivos** 4

**Justificación** 5

**Narrativa por capítulos** 6

**Índice** 8

**2. Métodos Tradicionales** 15

2.1. MRC (Método de la Ruta Crítica), CPM 16

2.2. PERT (Program Evaluation and Review Technique). 19

2.3. La programación 23

2.3.1. Cuantificación de los tiempos estimados 23

2.3.2. Inventario de recursos 23

2.3.3. Escala de tiempo en los pronósticos de duración 24

2.4. Representación Gráfica de un proceso 25

2.4.1. Barras Horizontales o Diagramas de Gantt. 25

2.4.2. Histogramas 29

2.4.3. Diagramas de flujo o Algoritmo 30

2.4.4. Matriz de secuencias 31

2.4.5. Reporte de estimaciones 32

**Contenido** **página**


**2. Descripción del Sistema. 33**

2.1. ¿Qué es un proyecto? 34

2.1.1. Definiciones 34

2.2. Antecedentes del Sistema de Estimación y Control de Proyectos 37

2.2.1. Factibilidad técnica 37

2.2.2. Factibilidad económica 37

2.2.3. Factibilidad operacional 38

2.3. Estándares 41

2.3.1. Estándares para códigos fuente 42

2.3.2. Control de cambios y nuevos productos 44

2.4. Descripción del Sistema de Estimación y Control de Proyectos 45

2.4.1. Módulos del Sistema de Estimación y Control de Proyectos 47

2.4.2. Módulo de Seguridad 48

2.4.3. Módulo de Básicos y Productos 50

2.4.4. Módulo de Partidas 51

2.4.5. Módulo de Inventarios 53

2.4.6. Módulo de Subcontratistas 54

2.4.7. Módulo de Estimaciones 55

2.4.8. Módulo de Estadísticas 56

2.4.9. Módulo de Infraestructura 57

2.4.10. Módulo de Guías Contabilizadoras 58

2.4.11. Módulo de Procesos Batch 65

**C o n t e n i d o** **págir**


**3. Análisis del sistema** **6**

3.1. Utilización de Modelos 6
3.2. Diagrama de Flujo de Datos (DFD) 6
3.3. El Proceso 7
3.4. El Flujo 7
3.5. El Archivo 7
3.6. La entidad externa o terminador 7
3.7. Guía para la construcción de un DFD 7
3.8. El Diccionario de Datos. 8
3.9. La necesidad de la notación en el Diccionario de Datos 8
3.10. Definiciones 8
3.11. Implantación del Diccionario de Datos. 8

**Contenido** **página**


**4. Diseño del sistema** **87**

4.1. Modelo Conceptual de Datos 91

4.2. Componentes del Modelo Entidad-Relación 91

4.2.1. Entidades 91

4.2.2. Relaciones 93

4.2.3. Grados de Relación 95

4.2.4. Matriz de Relaciones 96

4.2.5. Representación del Diagrama Entidad-Relación 97

4.2.6. Atributos 98

4.2.6.1 Opcionalidad de Atributos 98

4.2.6.2. Identificador de Atributos 99

4.3. Modelo Conceptual de Datos Avanzado 104

4.3.1. Normalizar el Modelo de Datos 104

4.3.1.1 regla de la Primera Forma Normal 105

4.3.1.2. Regla de la Segunda Forma Normal 105

4.3.1.3. Regla de la Tercera Forma Normal 106

4.4. Otros Modelos 108

4.4.1. Modelo Jerárquico de Datos 108

4.4.2. Modelo Recursivo 108

4.4.3. Modelo de Roles 108

4.4.4. Modelo de Subtitpos 109

4.4.5. Modelos de Relaciones Exclusivos 109

4.4.6. Modelos de Datos de Tiempo 109

4.5. Conceptos de Base de Datos Relacional 110

4.5.1. Llaves Primarias 111

4.5.2. Llaves Foráneas 111

4.5.3. Integridad de Datos 111

4.5.4. Constraints de Integridad de Datos 111

4.6. Diseño de la Base de Datos 113
4.6.1. Diseño Inicial de la Base de Datos 114
4.6.1.1.Mapear las Entidades 115
4.6.1.2.Mapear Atributos a Columnas 115
4.6.1.3.Mapear UIDS a Llaves Primarias 116
4.6.1.4.Mapear Relaciones Para Llaves Foráneas 116
4.6.1.5.Escoger Opciones de Arco 117
4.6.1.6.Escoger Opciones de Subtitpos 118
4.6.2. Normalización de Tablas 120
4.6.2.1.Reconocer Datos Sin Normalizar 121
4.6.2.2.Conversión a la Primera Forma Normal 121
4.6.2.3.Conversión a la Segunda Forma Normal 121
4.6.2.4.conversión a la Tercera Forma Normal 122
4.6.2.5.Especificar la Integridad Referencial 123
4.6.3. Diseño de Índices 124

**Contenido** **página**


**5. Desarrollo del sistema 124**

5.1. Implementación de la base de datos 125

5.1.1. Sistema de Manejo de Bases de Datos (DBMS) 127

5.1.2. Arquitectura de un sistema relacional de base de datos 129

5.2. Programación del Sistema 132

5.2.1. El proceso de traducción 132

5.2.2. Elección de un lenguaje 134

5.2.3. Clases de lenguajes 135

5.2.3.1. Primera Generación de lenguajes 136

5.2.3.2. Segunda Generación de Lenguajes 136

5.2.3.3. Tercera Generación de Lenguajes 137

5.2.3.4. Lenguajes de Cuarta Generación 138

5.3. Programación Orientada a Objetos (POO) 140

5.3.1. Objetos 141

5.3.2. Mensajes 143

5.3.3. Clases 144

5.3.4. Métodos de Organización 146

5.3.5. Herencia 148

5.4. Programación Orientada a Eventos 149

5.5. Como desarrollar el sistema de estimación y control de proyectos 150

**Contenido** **página**


**6. Pruebas y Puesta en Marcha** **152**

6.1. Pruebas del Sistema 153

6.1.1. Flujo de Información de la Prueba 153

6.1.2. Técnicas de Prueba 155

6.1.2.1. Prueba de la Caja Blanca 156

6.1.2.2. Prueba de la Caja Negra 157

6.1.3. Estrategia de Prueba del Software 158

6.1.4. Prueba de Unidad 160

6.1.5. Prueba de Integración 161

6.1.6. Pruebas de validación 162

6.1.7. Prueba del Sistema 163

6.2. Puesta en Marcha del Sistema 164

6.2.1. Capacitación a Usuarios 165

6.2.1.1. Capacitación en el uso del Sistema de E. y C. de Proyectos 166

**Conclusiones.** **169**

**Bibliografía.** **172**

# CAPÍTULO 1

# Métodos Tradicionales

**Es mejor utilizar nuestra propia cabeza**
**un par de minutos que un**
**par de días una computadora.**

**Francis Crick.**
**Biofísico y Premio Nobel Británico.**

# CAPITULO 1

# Métodos Tradicionales

## 1.1 .- MRC Método de la Ruta Crítica [*CPM Critical Path Method*].

En la planificación y control de los proyectos de investigación y desarrollo es común el utilizar el análisis de redes incluyendo PERT-CPM (Program Evaluation and Review Technique-Técnica de Revisión y Evaluación de Proyectos) y CPM (Critical Path Method-Método de la Ruta Crítica).

Una RED es una gráfica que consta de un conjunto de puntos de unión llamados nodos, siendo unidos ciertos pares de los nodos por medio de líneas llamadas ramas ( o "arcos", "eslabones" o "aristas", así se considera una red como una gráfica con un flujo de algún tipo en sus ramas.

Una cadena entre los nodos i y j es una sucesión de ramas que conectan a estos dos nodos.

Cuando se especifica la dirección en que se recorre la cadena, se le da el nombre de trayectoria.

Un ciclo es una cadena que une a un nodo consigo mismo sin retroceder sobre sus pasos.

Una gráfica conexa es aquella en la que existe una cadena que conecte a todo par de nodos.

Un árbol es una gráfica conexa que no contiene ciclos.

Uno de los teoremas de la teoría de gráficas afirma que una gráfica que contiene **n** nodos es conexa si tiene **(n-1)** ramas y ningún ciclo, tal gráfica es un árbol.

Se dice que la rama de una gráfica está orientada ( o dirigida) si existe un sentido atribuido a la rama, de modo que uno de los nodos se considera como el punto de origen y el otro como el punto de destino. Una gráfica orientada es una en la que todas las ramas están orientadas. Si una gráfica orientada es una red, la orientación de una rama es la dirección factible de flujo a lo largo de ésta. Sin embargo, no es necesario que una red esté orientada, porque puede ser factible que tenga flujo en cualquier dirección a lo largo de una rama. La capacidad de flujo de una rama es una dirección especificada es el límite superior para la magnitud factible del régimen de paso (o cantidad total de flujo) en la rama, en esa dirección. La capacidad de flujo puede ser cualquiera cantidad no negativa, incluyendo infinito.

Una rama está orientada si la capacidad de flujo es cero en una dirección. Un nodo también puede tener una capacidad de flujo limitada.

Un nodo en una red es una fuente si cada una de sus ramas tiene una orientación tal que el flujo se mueve hacia afuera del nodo. Análogamente, se conoce como nodo destino si cada una de sus ramas está orientada hacia ese nodo. Así entonces, las fuentes pueden imaginarse como los generadores del flujo y los destinos como los absorbentes de ese flujo.

La ruta más corta consiste en hallar la ruta más corta desde un origen hasta un destino, a través de una red que los conecta dada la distancia no negativa asociada con las ramas respectivas de la red.

**Algoritmo para el problema de la ruta más corta.**

Objetivo de la n-ésima iteración Hallar el n-ésimo nodo más cercano al origen. (Para ser repetido para n=1,2,..., hasta que el n-ésimo nodo más cercano es el destino.)

Entrada para la n-ésima iteración Los (n - 1) nodos más cercanos al origen (hallados en iteraciones anteriores), incluyendo su ruta más corta y su distancia al origen. (Estos nodos, más el origen, se llamarán nodos resueltos; los otros nodos no resueltos.)

Candidatos para el n-ésimo nodo más cercano Cada nodo resuelto que esté conectado directamente por medio de una rama a uno o más nodos no resueltos proporciona un candidato -- el nodo no resuelto con la rama de conexión más corta. (Los empates dan lugar a candidatos adicionales.)

Cálculo del n-ésimo nodo más cercano Para cada uno de tales nodos resueltos y su candidato, súmese la distancia entre ellos y la distancia de la ruta más corta desde el origen a este nodo resuelto. El candidato con esa distancia total más corta es el n-ésimo nodo más cercano (los empates dan lugar a nodos resueltos adicionales) y su ruta más corta es la que genera esta distancia.

Hasta aquí, el problema se ha descrito en términos de minimizar la distancia desde un origen hasta un destino. No obstante, en realidad el problema de redes que se está resolviendo es averiguar cuál trayectoria que conecta dos nodos especificados minimiza la suma de los valores de las ramas sobre la trayectoria. No hay razón alguna para que estos valores de las ramas necesariamente representen distancias, incluso de manera indirecta. Por ejemplo, las ramas podrían corresponder a actividades de alguna clase, en las que el valor asociado con cada rama es el costo de la actividad, otro valor asociado podría ser el tiempo o alguna otra cantidad.

## 1.2 .- PERT (Program Evaluation and Review Technique).

### Planeación y control del proyecto con PERT-CPM.

La dirección exitosa de proyectos a gran escala requiere de la planificación y coordinación de numerosas actividades interrelacionadas.

Con el fin de prestar auxilio en estas tareas, se han desarrollado desde los últimos años de la década de los cincuenta procedimientos formales, basados en el uso de redes y técnicas de redes. Los más prominentes de estos procedimientos han sido el PERT (Program Evaluation and Review Technique - Técnica de Revisión y Evaluación del Programa) y el CPM ( Critical Path Method - Método de la Ruta Crítica.)

Aunque tienen muchas variantes con nombres diferentes. Sin embargo la tendencia actual ha sido fusionar los dos procedimientos en lo que comúnmente se conoce como sistema tipo PERT.

No obstante que la aplicación original de los sistemas tipo PERT fue para evaluar la secuencia de un programa de investigación y desarrollo, también se está usando para medir y controlar el progreso en muchos otros tipos de proyectos especiales.

Ejemplos de estos tipos de proyectos incluyen programas de construcción, desarrollo de programas de computadoras, preparación de ofertas y proposiciones, planificación del mantenimiento y la instalación de sistemas de computadoras. Se ha aplicado esta clase de procedimiento para la producción de películas, campañas políticas y cirugía compleja.

Se diseña un sistema tipo PERT para ayudar en la planificación y el control, de modo que es posible que no comprenda mucha optimización directa. A veces uno de los objetivos principales es determinar la probabilidad de satisfacer límites de tiempo especificados. También identifica las actividades que tienen más probabilidad de convertirse en cuellos de botella y en las que, por lo tanto, debe programarse el esfuerzo máximo. Un tercer objetivo es evaluar el efecto de los cambios en el programa. Por ejemplo, evaluar el efecto de un desplazamiento contemplado de recursos, de las actividades menos críticas a las identificadas como cuellos de botella probables. También pueden ser evaluadas otras combinaciones de recursos y rendimientos. Otro uso importante es evaluar el efecto de las desviaciones respecto del programa.

Todos los tipos PERT usan una RED DEL PROYECTO para retratar gráficamente las interrelaciones entre los elementos de un proyecto. Esta representación como red del plan de un proyecto muestra todas las relaciones de precedencia referentes al orden en el que deben efectuarse las tareas.

En la terminología del PERT, cada rama de la red del proyecto representa una actividad, que es una de las tareas requeridas por el proyecto. Cada nodo representa un evento que comúnmente se define como el punto en el tiempo cuando todas las actividades conducen hacia ese nodo se completa. Las puntas de flecha indican las sucesiones en las que deben lograrse los eventos. Además, un evento debe preceder a la iniciación de las actividades que parten de ese nodo. (En realidad, a menudo es posible traslapar fases sucesivas de un proyecto, de modo que la red puede representar una idealización aproximada del plan del proyecto.) El nodo hacia el cual conducen todas las actividades (el destino de la red) es el evento correspondiente a la compleción del proyecto actualmente planeado.

La red puede representar el plan del proyecto desde su principio, o bien, si el proyecto ya ha sido iniciado, el plan para su compleción. En el último caso, cada fuente de la red representa el evento de continuar una actividad actual, o bien, el evento de iniciar una nueva actividad que pueda arrancarse en cualquier instante.

Las flechas a trazos, llamadas actividades ficticias, muestran únicamente relaciones de precedencia; no representan actividades reales. Una regla para construir estas redes de proyectos, es que pueden conectarse directamente dos nodos por más de una rama. También pueden usarse actividades ficticias para evitar que se viole esta regla cuando se tiene dos o más actividades concurrentes.

Después de desarrollar la red para un proyecto, el paso siguiente es estimar el tiempo requerido para cada una de las actividades. Estos tiempos se usan para calcular dos cantidades básicas para cada evento, a saber, su tiempo más anticipado y su último tiempo.
El tiempo más anticipado para un evento es el tiempo (estimado) en el cual ocurrir el evento si las actividades precedentes se inician tan pronto como es posible.

El último tiempo para un évento es el último tiempo (estimado) más allá de su tiempo más anticipado, en el que puede ocurrir un evento sin retrasar la compleción del proyecto.

La holgura para un evento es la diferencia entre su último tiempo y su tiempo más anticipado. La holgura para una actividad (i,j) es la diferencia entre (el último tiempo para el evento j) y (el tiempo más anticipado del evento i más el tiempo estimado para la actividad).

Los tiempos más anticipados se obtienen haciendo un paso hacia adelante a través de la red, empezando con los eventos iniciales y trabajando hacia adelante en el tiempo, hacia los eventos finales; calculando sucesivamente el tiempo en el que ocurrirá cada evento, si cada uno de los eventos inmediatamente precedentes ocurre en su tiempo más anticipado y cada una de las actividades que intervienen sólo consumen sus tiempos estimados.

Los últimos tiempos se obtienen sucesivamente para los eventos, realizando un paso hacia atrás a través de la red, empezando con los eventos finales y trabajando hacia atrás en el tiempo, hacia los eventos iniciales, calculando en cada vez el tiempo final en que puede ocurrir el evento, si cada uno de los eventos inmediatos siguientes ocurre en su último tiempo y cada actividad que interviene sólo consume su tiempo estimado.

Así, la holgura para un evento indica cuánto retraso en alcanzar el evento puede tolerarse, sin retrasar la compleción o termino del proyecto, y la holgura para una actividad indica lo mismo referente a un retraso en el termino de esa actividad.

Una ruta crítica para un proyecto es una trayectoria a través de la red tal que las actividades sobre ella tienen holgura cero. (Todas las actividades y eventos que tiene holgura cero deben encontrarse sobre una ruta crítica, pero ningunas otras pueden estarlo.)

Esta información acerca del tiempo más anticipado y el último tiempo, la holgura y la ruta crítica es inapreciable para el director del proyecto, a fin de determinar en dónde debe realizarse un esfuerzo especial a fin de mantenerse dentro del programa y valorar el impacto de las modificaciones del programa.

La estimación del tiempo hasta aqui parece ser que, sólo se ha supuesto para cada una de las actividades del proyecto. Pero existe una considerable incertidumbre acerca de cual será el tiempo.

La realidad es que el tiempo es una variable aleatoria que tiene cierta distribución de probabilidad. Por tal razón, la versión original del PERT suponía, el uso de tres tipos diferentes de estimaciones del tiempo, de la actividad para obtener una información básica respecto a su distribución de probabilidad. Entonces se usa esta información para todos los tiempos de las actividades para estimar la PROBABILIDAD de completar el proyecto para la fecha programada.

Las tres estimaciones del tiempo usadas en el PERT son:

- Una estimación más probable.
- Una estimación optimista.
- Una estimación pesimista.

Se debe tratar de que la estimación más probable (m), sea la estimación más realista del tiempo que podría consumir la actividad.

En términos estadísticos, es una estimación de la moda (el punto más alto) de la distribución de probabilidad para el tiempo de la actividad.

La estimación optimista (a) se pretende que sea el tiempo improbable pero posible si todo marcha bien. Es en esencia estadística, una estimación de la cota inferior de la distribución de probabilidad.

La estimación pesimista (b) sea el tiempo improbable pero posible si todo marcha mal. Es en esencia estadística una estimación de la cota superior de la distribución de probabilidad.

Se hacen dos suposiciones para convertir m, a y b en estimaciones del valor esperado y la variancia del tiempo transcurrido que requiere la actividad. Una es que la desviación estándar (raiz cuadrada de la variancia), es igual a un sexto del intervalo de requerimientos de tiempo razonablemente posibles; es decir, es la estimación deseada de la variancia. El razonamiento es que se considera que las colas de muchas distribuciones de probabilidad (como la distribución normal) se encuentran aproximadamente a 3 desviaciones estándar de la media, de modo que habría una extensión de aproximadamente 6 desviaciones estándar entre las colas.

Para obtener el valor esperado estimado, se requiere suponer acerca de la distribución de probabilidad del tiempo requerido por la actividad, ésta es que la distribución es aproximadamente del tipo $\beta\varepsilon\tau\alpha$, en la que la moda es **m**, la cota inferior es **a** , la cota superior es **b** y la desviación estándar es $\Gamma$.

$$\Gamma = ( b - a ) / 6$$

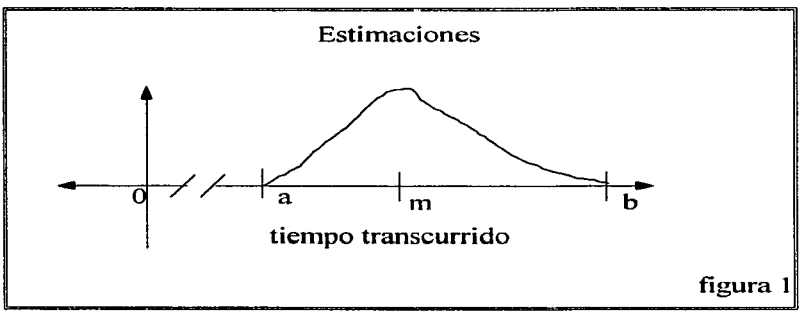



figura 1

## 1.3 La programación.

### 1.3.1 Cuantificación de los tiempos estimados.

Dentro de la metodología de estimación, programación y control de proyectos, el paso llamado de programación, consiste en fijar cuantitativamente duraciones estimadas a cada una de las actividades que forma la estructura del proyecto; es un pronóstico numérico derivado de los recursos y medios al alcance, o de aquellos que deberán asignarse para cumplir con el plan original.

En este proceso de programación, se entiende por recurso a todo elemento que puede ser útil como medio para el logro de un objetivo.

De todos ellos, el tiempo es el factor que más interesa a toda programación. Ello debido a que se considera al tiempo como el común denominador de todos los otros recursos.

Dentro del mercado el producto u objetivo no dependen exclusivamente del tiempo sino de todos los factores que para su logro intervienen. El dinero y todos los demás recursos que intervienen son valorizados por el programador en su potencial de trabajo, para medirse en unidades de consumo de tiempo. Pero lo que si es vital es encontrar los recursos críticos, para atacarlos y lograr el fin deseado sin obstáculos.

### 1.3.2.Inventario de recursos.

Antes de cuantificar los tiempos, es necesario levantar un inventario de medios de que disponemos para cada actividad. Del mismo modo es fundamental asesorarnos de elementos técnicos que conozcan a fondo la materia que constituye la fuente de los datos que influirán en los pronósticos. Debemos

además tener la certeza de esta información sea suficientemente amplia para establecer una estadística confiable, completa y reciente.

Es de suma importancia informarnos acerca de si esos datos obtenidos son aplicables al lugar, clima, época del año u otros factores regionales que cambiarán nuestros pronósticos.

Naturalmente debemos acudir a nuestro archivo, si lo tenemos, o en cualquier forma asegurarnos acerca de los datos más recientes de tiempo y costos de proyectos, de obras, trabajos, procesos o actividades del mismo género del que tratamos de pronosticar. de lo completo del acervo de esta información, dependerá en gran parte el éxito o el fracaso de nuestro programa, ya sea con respecto al tiempo involucrado, o a la inversión presupuestada.

Es parte de un ciclo dinámico, el incluir dentro del sistema la formación de un archivo de experiencias, que nos guiará en el futuro.

Este archivo o base de datos o sistema de información puede ser creado mediante técnicas estadísticas, auxiliándose de paquetes el en mercado, no cito alguno de ellos por ser abundantes y los hay desde calculadoras preprogramadas, hasta sistemas para main frame.

### 1.3.3 Escala de tiempo en los pronósticos de duración.

La escala de los tiempos por pronosticar evidentemente la debemos expresar en segundo, minuto, hora, día, semana, mes, año, lustro y demás unidades submúltiplos o múltiplos de tiempo. El módulo o patrón de comparación que nos conviene utilizar lo determinará el tipo de trabajo a realizar. Por ejemplo para la construcción de una presa hidroeléctrica, que el proyecto en si durará un par de años o más, bastará escoger la semana como unidad práctica. Sin embargo al formular un programa es ilógico mezclar unidades de tiempo fuera de proporción, como segundos y semanas.

## 1.4 .- Representación Gráfica de un proceso.

### 1.4.1.- Barras Horizontales o Diagramas de Gantt.

En la planeación y control de proyectos es muy común el uso de formas preimpresas en las cuales, se presentan las tareas específicas de cada proyecto. Son auxiliares para tener a la vista el proyecto y sus partes.

Ejemplo de estas formas son los Diagramas de Gantt, que no son otra cosa que un listado de actividades, en la que se asigna un responsable o responsables (recursos), el tiempo y costo estimado de cada tarea.

Cada actividad es agrupada en un partida o subproyecto y a la vez estos en el proyecto mismo.

Cada tarea ocupa dos renglones el primero es utilizado para el estimado, el otro es usado con el real, de esta manera es posible visualizar el estimado contra el real.

El diagrama de Gantt puede ser utilizado como un reporte de avance, y es definido como una representación gráfica de las diferentes fases o etapas que comprende un proyecto.

**La gráfica de Gantt refleja la siguiente información:**

- Tiempo planeado para terminar cada fase.
- Porcentaje de avance real de casa fase.
- Tiempo transcurrido de todo el proyecto.
- Porcentaje de avance del proyecto.
- Fechas de corte.
- Los productos necesarios para terminar la fase.

El uso más común del Gantt es mantener informado al usuario y a los demás interesados sobre el avance del proyecto.

El reto a vencer en todo lo que supone planeación es que el experto que asigna los tiempos, debe contar con información precisa y cuantificable, para con ello evitar los dos posibles errores de una estimación.

**Errores de estimación.**

*1. Estimar menos del real*
*2. Estimar más del real.*

Para el primer caso el responsable no terminará a tiempo, con lo que es proyecto sufrirá un retraso, y en el segundo la holgura será pérdida en la utilización del recurso, en ambos casos esto implica un aumento en los costos.

Para el caso de un proyecto como es el desarrollo de sistemas informáticos, la experiencia o sentimiento del experto para asignar el tiempo estimado más real, no es suficiente para garantizar que el tiempo estimado y el tiempo real sean iguales. Por ello es conveniente utilizar herramientas estadísticas, y generar patrones para medir y cuantificar, los tiempos y recursos necesarios para cada una de las tareas.

Sugiero para este caso en particular definir con la mayor exactitud posible el tipo de recurso y el tipo producto, ya que se incurre en el error de desear cuantificar la actividad. No es factible decir que el análisis de un programa es cuantificable, pero el documento del análisis si lo es.

En el caso de un programa, debido a la diversidad de estos, es necesario clasificarlos según el producto que el mismo genera y, las operaciones que en el intervienen, tanto para su análisis, desarrollo y puesta en marcha, así cómo el tipo de profesional que lo esta creando.

Ya que un experto puede llegar al producto deseado con menor tiempo, que un novato. Pero el costo unitario de tiempo del experto es mucho mayor que el de un novato, así habrá que valuar cual de ellos es el óptimo. Si bien en algunas ocasiones el tiempo es imperante, en otras lo es el costo.

Para el caso de retroalimentar un proyecto, debido a la detección de un retraso, para la situación de modificar la asignación de recursos humanos. Es necesario recordar que:

***"...Se requiere de una madre y 9 meses para gestar un niño, y 9 madres juntas NO logran generar un niño en un mes..."***

Supongamos que una empresa cuenta con varios equipos de trabajo especializados, por ejemplo: una constructora de inmuebles, tiene un equipo de diseño, de informática, de contabilidad, de subcontrataciones, de compras, demoliciones, peritajes, mensajería, de recursos humanos y, de administración general.

Supongamos además que es mediante un concurso abierto el adjudicarse la construcción del proyecto.

La mecánica desde el inicio hasta el producto terminado es:

- Inscribirse para el concurso.
- Levantamientos de información acerca del terreno de mismo.
- Diseño arquitectónico y de ingeniería civil.
- Análisis de planos para determinar los subproyectos, partidas y tareas propias del proyecto; así como los materiales, productos, mano de obra, herramientas y otros insumos necesarios.
- Estimar tiempos y costos de cada tarea.
- Generar el anteproyecto y presentar el plan estimado de obra.
- Teniendo la adjudicación del contrato para construir el inmueble.

- Conseguir los permisos gubernamentales para su construcción.
- Iniciar según los planes previstos y arrancar con el control de obra.

Conjuntamente interactuan los sistemas administrativos, como son: nómina, cuentas por pagar, cuentas por cobrar, inventarios, sistemas de control de créditos, y los sistemas de diseño como son el Diseño Auxiliado por Computadora (CAD), y los sistemas de Diseño Auxiliado por Computadora con Diseño Mecánico Auxiliado por Computadora (CAD-CAM).

Ejemplo de un diagrama de Gantt es el de la ***figura 1.4.1***

## *DIAGRAMA DE GANTT*

Responsable: ***Julio López García*** **Semana del:** *960805* **al:** *960809*

| RESPONSABLE A | B | C | D | ACTIVIDADES | | L | M | M | J | V | S | D | L | AVANCE % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | | | | REPORTE DE ESTIMACIONES POR PROYECTO | E | | | | | | | | | 100 |
| | | | | | R | | | | | | | | | 100 |
| | X | | | REPORTE DE ESTIMACIONES POR RESPONSABLE | E | | | | | | | | | 100 |
| | | | | | R | | | | | | | | | 100 |
| | | X | | REPORTE DE ESTIMACIONES POR RANGO DE FECHAS | E | | | | | | | | | 100 |
| | | | | | R | | | | | | | | | 100 |
| | | | X | REPORTE DE PRODUCTOS TERMINADOS | E | | | | | | | | | 100 |
| | | | | | R | | | | | | | | | 0 |
| | | X | | REPORTE DE AVANCE REAL | E | | | | | | | | | |
| | | | | | R | | | | | | | | | |
| X | | | | REPORTE DE AVANCE PROGRAMADO | E | | | | | | | | | 100 |
| | | | | | R | | | | | | | | | 100 |

A = JULIO LÓPEZ GARCÍA B = FELIPE PÉREZ ACOSTA
C = RICARDO ROJAS OVIEDO D = TOMAS RUIZ RUIZ

***Figura 1.4.1..Diagrama de Gantt.***

### 1.4.2.- Histogramas.

También conocido como diagrama de barras verticales, consiste de barras verticales congruentes con una escala bidimensional donde se combinan dos elementos produciendo lo que se denomina un histograma; para nuestro propósito, es relacionar el tiempo contra los recursos. (***ver figura 1.4.2***).

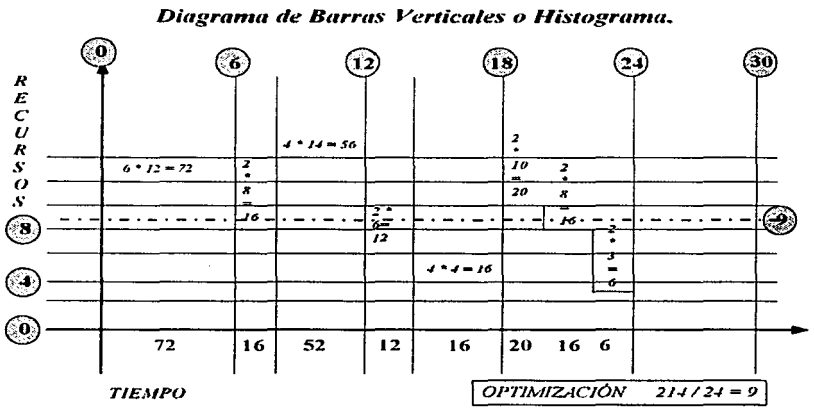



*Figura 1.4.2*

### 1.4.3.- Diagramas de flujo o Algoritmo.

El algoritmo es un paso que consiste en una serie muy definida de instrucciones que resuelve un problema (plan). Formulado por el programador, su representación gráfica utiliza signos convencionales de las operaciones involucradas ( alimentación de datos, alternativas de decisión, iteración, etc.) la ***figura 1.4.3*** es un ejemplo muy simple de lo que es un diagrama de flujo.

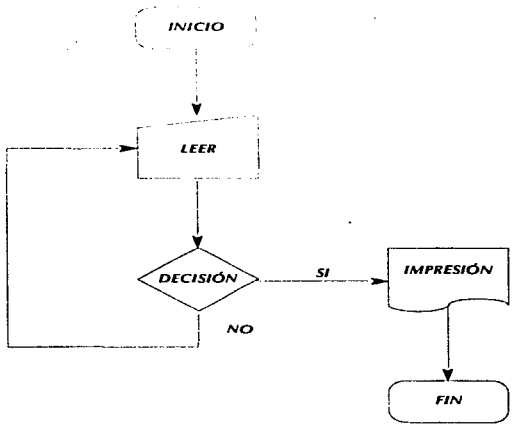



***Figura 1.4.3 Ejemplo de Diagrama de flujo***

### 1.4.4.- Matriz de secuencias.

En cierta manera se puede considerar como un algoritmo, expresado en forma tabular, en vez de diagrama de flujo. La iniciación, la secuencia y la terminación de un proyecto se interpretan claramente a través de esta matriz. La ***figura 1.4.4*** representa una síntesis de los pasos involucrados en fabricar una puerta y colocarla en obra.

La matriz se interpreta de la siguiente manera:

Los círculos se han designado como signo convencional para representar el principio de un proyecto total; al existir dos de ellos en la tabla, inferimos que existe una iniciación simultánea de dos actividades. Frente a las diversas actividades listadas, cruces que le siguen marcan la secuencia deseada del proceso. Finalmente, los triángulos o "mojoneras" señalan la terminación del plan o sea la culminación del objetivo.

| ACTIVIDAD | CLAVE | DURACIÓN [hora] | ab | be | ed | cd | df | fg | gh | hi |
|---|---|---|---|---|---|---|---|---|---|---|
| Hacer bastidor | ab | 4 | ○ | ✦ | | | | | | |
| Forrar bastidor | be | 2 | | | ✦ | | | | | |
| Embroquelar | ed | 2 | | | | | ✦ | | | |
| Hacer cajón | cd | 3 | | | ○ | ✦ | | | | |
| Fijar hoja al cajón | df | 6 | | | | | ✦ | | | |
| Colocación en obra | fg | 2 | | | | | | ✦ | | |
| Colocación de chambrana | gh | 3 | | | | | | | ✦ | |
| Colocación de picaporte | hi | 2 | | | | | | | | △ |

***Figura 1.4.4 Matriz de secuencias.***

### 1.4.5.- Reporte de estimaciones.

Al usar computadoras en un proyecto, las salidas generadas , por medio de pantallas (consultas en línea) o por impresiones, conocidas como reportes; son muy empleadas por los interesados en control de un proyecto, la ***figura 1.4.5*** , es un posible ejemplo de lo que sería uno de estos reportes. En el que puede observarse el nombre de la empresa, título del reporte, número de hoja, nombre del proyecto, fecha de impresión, fecha de proceso de la información, códigos y descripciones del subproyecto(s) , partida(s), concepto(s), porcentajes, comparativos, unidades de medida, e importes.

**ACME** **HOJA:**
**0009**

**Control Avance de Proyecto.**

**PROYECTO** **Vie. 13 May, 2008.** **Sin fecha de**
**proceso.**

***CÓDIGO*** ***DESCRIPCIÓN***

**01** **COMERCIAL CASA DE BOLSA**
**61** **COMERCIAL CASA DE BOLSA**
**02** **SEGUNDA ETAPA**

| CÓDIGO | DESCRIPCIÓN | U M | CANTIDAD | ACUMULAD ANTERIOR | ESTIMADO | ACUMULAD. ACTUAL | EXTRAS | ADITIVAS | POR EJERCER | CVE |
|---|---|---|---|---|---|---|---|---|---|---|
| 01610260 | ARMADURA TIPO A-6 | PZA | 22.00 | 10.00 | 0.00 | 10.00 | 0.00 | 0.00 | 6.00 | U |
| | | | 6053346.41 | 11073598.02 | 0063626.56 | 0063626.56 | 0.00 | 0.00 | 3020672.46 | $ |
| | | | | 100.0000 | 72.7273 | 0.00 | 72.7273 | 0.00 | 0.00 | 27.2727 | % |
| 01610280 | METAL DESPLEGADO EST | m2 | 436.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 436.00 | U |
| | | | 19891.66 | 0623372.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0623372.10 | $ |
| | | | | 100.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 100.0000 | % |
| 01610290 | ACARREO DE ESCOMBRO | m3 | 500.00 | 440.00 | 0.00 | 440.00 | 0.00 | 0.00 | 60.00 | U |
| | | | 15826.34 | 7963179.00 | 7007699.60 | 7007699.60 | 0.00 | 0.00 | 956680.40 | $ |
| | | | | 100.0000 | 88.0000 | 0.0000 | 88.0000 | 0.0000 | 12.0000 | % |
| Subtotal | SEGUNDA ETAPA | | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | U |
| | | | 3633336618.2 | 2163120836.4 | 0.00 | 2163120836.4 | 15504498.01 | 0.00 | | $ |
| | | | | 60.0373 | 0.00 | 60.0373 | 0.00 | 0.00 | 39.9627 | % |
| Subtotal | COMERCIAL CASA DE BOLSA | | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | U |
| | | | 3633336618.2 | 2163120836.4 | 0.00 | 2163120836.4 | 15504498.01 | 0.00 | | $ |
| | | | | 60.0373 | 0.00 | 60.0373 | 0.00 | 0.00 | 39.9627 | % |
| Total | COMERCIAL CASA DE BOLSA | | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | U |
| | | | 3633336618.2 | 2163120836.4 | 0.00 | 2163120836.4 | 15504498.01 | 0.00 | | $ |
| | | | | 60.0373 | 0.00 | 60.0373 | 0.00 | 0.00 | 39.9627 | % |

***Figura. 1.4.5 Reporte de Control Avance de Proyecto.***

Ω

# CAPÍTULO 2

## Descripción del Sistema.

*" Prefiero sufrir una injusticia, y no cometerla."*

*Sócrates.*

*El que no sabe, y no sabe que no sabe, es un necio; evítalo.*
*El que no sabe y sabe que no sabe es un ignorante; enséñale.*
*El que sabe y no sabe que sabe está dormido; despiértale.*
*El que sabe y sabe que sabe es un sabio; síguele.*

*San Mateo, Apóstol.*

# CAPITULO 2
# Descripción del Sistema.

## 2.1.- Qué es un Proyecto?.

### 2.1.1.- Definiciones:

***Proyecto es idea que se tiene de algo que se piensa hacer y de cómo hacerlo.***

***Proyecto es conjunto de escritos , cálculos y dibujos que se hacen para dar idea de cómo ha de ser y lo que ha de costar una obra.***

***Proyecto es un conjunto de productos bien definidos, encaminados a producir un resultado en un período de tiempo marcado.***

***El proyectar es idear, trazar, disponer o proponer el plan y los medios para la ejecución de una cosa.***

Por ejemplo el proyecto para desarrollar un sistema de información está formado por los siguientes subproyectos o elementos.

1. Investigación preliminar.
2. Determinación de los requerimientos del sistema.
3. Diseño del sistema.
4. Desarrollo de software.
5. Prueba de los sistemas.
6. Implantación y evaluación.

Otro ejemplo de proyecto "...Construcción de un edificio Multifamiliar..." está formado por los siguientes subproyectos:

1. Levantamientos de información del campo.
2. Diseño arquitectónico.
3. Diseño estructural.
4. Diseño de instalaciones y servicios.
5. Generación del plan de obra.
6. Limpieza y nivelación del terreno.
7. Cimentación.
8. Construcción de cisternas.
9. Construcción de drenajes.
10. Levantamiento de estructuras.
11. Muros y losas.
12. Instalaciones hidrosanitarias.
13. Instalaciones eléctricas.
14. Instalaciones de aire acondicionado.
15. Jardines y áreas de proindiviso.
16. Acabados en muros y techos.

Cabe citar que, para lograr el producto final en este proyecto, interactúa el sistema de control de proyectos con otros sistemas, como el de inventarios, control de

subcontratistas, contabilidad, compras, cuentas por pagar y demás afines al caso, ello depende de la magnitud y características propias del proyecto.

A su vez los subproyectos, deben ser desglosados en partidas, las cuales están compuestas por tareas específicas, relacionadas por fuerza a un producto.
El producto o tarea es la unidad mínima para la planeación y control.
Los productos tiene las siguientes cualidades:

Tangibilidad .- Resultado concreto, no la actividad que lo produce.

Responsable .- El producto debe ser obtenido por una sola persona, o por un sólo equipo.

Magnitud del esfuerzo requerido .- Es el tiempo requerido para obtener un producto. Este no debe ser mayor al tiempo total entre el tiempo de ciclo de cada operación de control. De esta manera se asegura que un recurso reporte al menos un producto a la semana y sea reflejado en el avance del proyecto y en la productividad del recurso.

Criterio de Terminación .- Es un valor binario ( 1 o 0) de SI o NO está terminado el producto.

## 2.2.- Antecedentes del Sistema de Estimación y Control de Proyectos.

Dentro del ciclo de vida clásico del desarrollo de sistemas, el primer punto es la investigación preliminar, aunque no existe ningún método correcto para desarrollar un sistema de información, sí existen diferentes formas para producir el sistema correcto para una aplicación.
El indicador definitivo del éxito de un método de desarrollo en particular, es aquel que se refiere a los resultados obtenidos y no a la "precisión" teórica del método.

Este sistema pretende resolver un problema, controlar un proyecto, otra razón es aprovechar una oportunidad , generando un cambio para ampliar o mejorar el rendimiento económico de la empresa y su combatividad.

Dentro del estudio preliminar es muy importante el Estudio de Factibilidad, ya que de este se desprende si el sistema puede lograrse o no. A la vez el estudio de factibilidad depende de cada caso en particular, ya que básicamente es necesario responder a tres cuestionamientos que son:

### 2.2.1.- Factibilidad técnica.

¿Puede realizarse el proyecto nuevo con el equipo actual, la tecnología existente de software y el personal disponible?

### 2.2.2.- Factibilidad económica.

Al crear el sistema, ¿los beneficios que se obtienen serán suficientes para aceptar los costos?, ¿los costos asociados con la decisión de NO crear el sistema son tan grandes que se debe aceptar el proyecto?

### 2.2.3.- Factibilidad operacional.

Si se desarrolla e implanta, ¿será utilizado el sistema?, ¿existirá cierta resistencia al cambio por parte de los usuarios, que dé como resultado una disminución de los posibles beneficios de la aplicación?

Nuestro reto a vencer es ¿Cómo estimar un proyecto y cómo controlar dicho proyecto?, para ello debemos contestar las siguientes preguntas:

**¿Qué es lo que voy hacer?**
R.- Estimar y controlar un proyecto.

**¿Cómo lo voy hacer?**
R.- Mediante un sistema de información para el procesamiento de transacciones.

**¿Con qué frecuencia?**
R.- De acuerdo con el ciclo del proyecto.

**¿Qué tan grande es el volumen de transacciones o de decisiones?**
R.- Relacionado con el tamaño del proyecto.

**¿Cuál es el grado de eficiencia con el que se efectúan las tareas?**
R.- Se intenta que sea acorde con las necesidades.

Nuestro sistema de estimación y control de proyectos cae dentro de los sistemas para el procesamiento de transacciones, ya que tiene como finalidad mejorar las actividades rutinarias de una empresa y de las que depende toda la organización.

El procesamiento de transacciones, que es el conjunto de procedimientos para el manejo de éstas, incluye entre otras, las siguientes actividades:

- Cálculos.
- Clasificación.
- Ordenamiento.
- Almacenamiento y recuperación.
- Generación de resúmenes.

El gran volumen de transacciones precisas asociado con el nivel operativo de una organización, junto con la capacidad de los administradores para desarrollar procedimientos específicos para manejarlos, conduce con bastante frecuencia a la implantación de ayuda asistida por computadora. Los analistas deben diseñan tanto los sistemas como los procesos para el manejo de actividades.

La *fig. 2.1.1* representa la relación entre sistemas de información y los niveles de una organización.

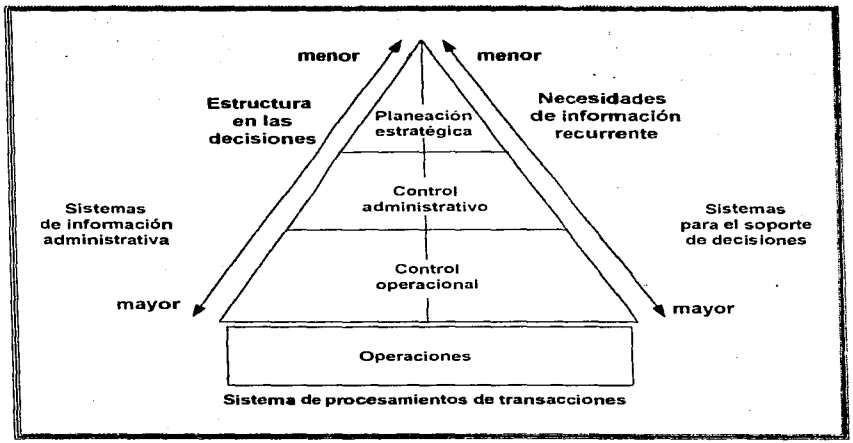



fig. 2.1.1

## 2.3.- Estándares.

Honorato de Balzac con un grupo de amigos escribían obras de teatro, pero era tal la falta de comunicación entre los mismos, que el resultado era una obra que metafóricamente en el primer acto los tres cochinitos representaban a los tres mosqueteros, en el segundo acto los tres mosqueteros interpretaban a los tres ositos y en el tercer acto, Don Quijote lloraba la muerte de Superman.

En los equipos de trabajo es importante tener acuerdos de la forma en que se van a cumplir los objetivos, de esta manera el esfuerzo individual, se integra correctamente. De lo contrario existe una competencia interna, que no permitirá que el equipo logre la consolidación del trabajo realizado.

Los proyectos de desarrollo de sistemas implican varias etapas desde su concepción hasta su explotación y pueden durar lapsos grandes e involucrar a muchos individuos. Los convenios o estándares que el director del proyecto implante son precisamente para coordinar los esfuerzos de cada elemento del equipo.

Con los estándares obtenemos beneficios tales como:

1. Puntos de referencia claros en la estructura del proyecto.
2. Compatibilidad con cada uno de los elementos.
3. Crear patrones de comparación.
4. Evitar duplicidad de rutinas comunes.
5. Consistencia en el producto generado.
6. Los usuarios son beneficiados en el aprendizaje del sistema.
7. Facilidades para propios y ajenos en el conocimiento del proyecto.
8. Facilidades para cubrir ausencias y evitar los cuellos de botella.

Al observar un partido de fútbol entre dos bandos, se distingue claramente el concepto de equipo.

En el equipo cada elemento participa con una actividad bien definida, por ejemplo el portero, el defensa, el medio volante, el delantero.

Los grupos son conjuntos de elementos con una característica en común por ejemplo: el grupo de los números enteros, o el grupo de las manzanas rojas.

La diferencia entre equipo y grupo, es que en el equipo cada uno de los elementos tiene una actividad bien definida y en el grupo sólo característica común, para ser agrupado.

### 2.3.1.- Estándares para códigos fuente (Programas, Subrutinas, Reportes)

La nomenclatura de un código fuente dentro de un sistema de información computarizado, se determino según la experiencia en el manejo de estos equipos, ya que el nombre debe empezar con cualquier carácter y por lo general el común denominador para la longitud es de 8 campos o dígitos, de los cuales el primero debe ser un carácter alfanumérico y los demás, números o letras. En el caso de computadoras personales existe una extensión de este nombre que es delimitado por un punto y se compone de tres caracteres alfanuméricos, los cuales definen el contenido del archivo por ejemplo la extensión **TXT** es muy empleada para archivos que contienen texto la extensión prg para los que tiene código fuente, la extensión **cdx** indica que es un archivo que contiene índices, DBF contiene datos en formato base de datos (Data Base Format), es otro tema la definición total de estas extensiones.

La figura 2.3.1 expone el estándar para la nomenclatura de archivos con código fuente del sistema de Control Maestro de Proyectos.

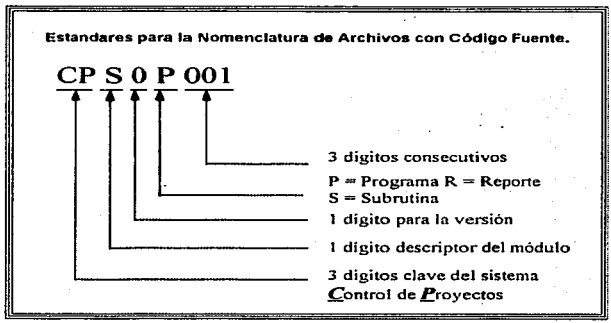



*figura 2.3.1*

La figura 2.3.2 expone el estándar para la nomenclatura de archivos de datos.

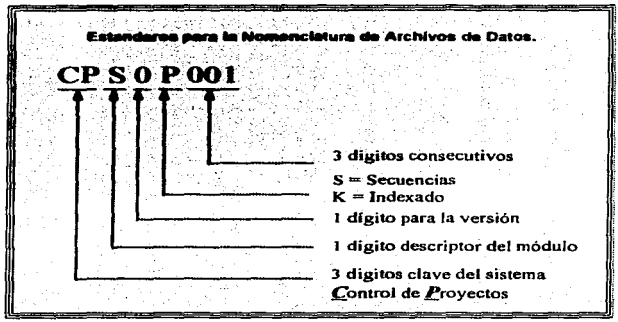



***figura 2.3.2***

Es importante que exista independencia entre datos y programas.

### 2.3.2.- Control de cambios y nuevos productos.

El ciclo de vida de un sistema es similar al de cualquier individuo u organización, nace, evoluciona, se reproduce y muere, por lo tanto antes de concebir alguna "creación", es muy importante tener una idea clara de lo que se desea, ya que una vez que se le da vida, habrá que ayudarla a su sano desarrollo, hacerla productiva y alargar todo lo posible su existencia.

La creación de un sistema, es un proceso en el que regularmente, intervienen distintas entidades, vinculadas precisamente por la necesidad de contar con un apoyo para simplificar sus operaciones, optimar la calidad de sus productos, ofrecer un mejor servicio a sus clientes e incrementar la productividad de sus recursos.

Al desarrollar un sistema, generalmente pensamos en obtener más de uno de los beneficios antes descritos y por lo tanto estamos conscientes que debemos invertir los recursos necesarios para alcanzar los objetivos planteados y que una vez que nace nuestro sistema, estamos obligados a mantenerlo en forma para evitar en lo posible su obsolescencia y así nos redituе los dividendos que de el esperamos.

En la práctica la generación de un sistema que contemple "TODO" es utópico y que siempre será mejor el 20% de "ALGO" que el 100% de "NADA", desde luego que ese "ALGO" debe ser un producto parcial del "TODO" y deberá representar un beneficio para quien lo utilice. De tal manera podemos disponer de productos parciales por etapas predefinido, y ello nos permitirá:

1. Recopilar las experiencias de los involucrados en la generación del Sistema.
2. Optimar sistemáticamente los procesos que así lo requieran.
3. Responder a las nuevas necesidades dadas por la dinámica del cambio.
4. Proporcionar resultados en períodos relativamente cortos.

El control de cambios y nuevos módulos plantea los siguientes objetivos:

1. Contar con una herramienta para el control y evolución del sistema.
2. Proporcionar productos debidamente probados.
3. Garantizar una disponibilidad continúa del sistema en su funcionamiento.
4. Contar con herramientas de respaldo para dar respuesta a eventualidades.

Incrementar la productividad de los recursos físicos y humanos.

## 2.4.- Descripción del Sistema de Estimación y Control de Proyectos.

### Sistema de Estimación y Control de Proyectos.

*Diagrama Jerárquico.*

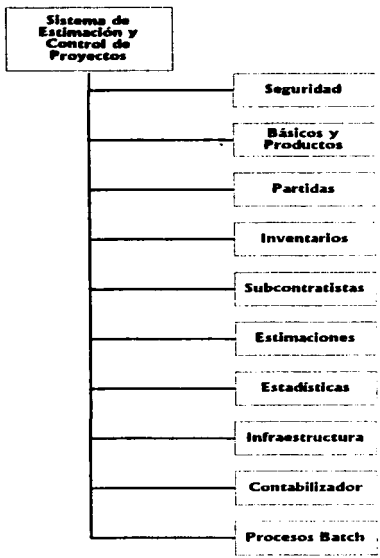
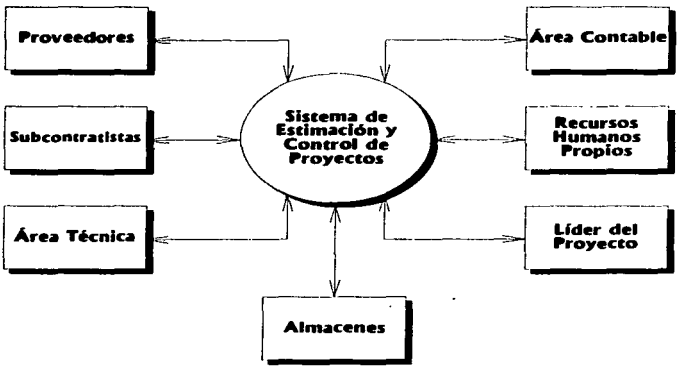



*Figura 2.4.1*

**Sistema de Estimación y Control de Proyectos.**

# *Diagrama de Contexto.*



## 2.4.1.- Módulos del Sistema de Estimación y Control de Proyectos.

***1.- Módulo de Seguridad.***
***Objetivo:*** Salvaguardar la integridad del sistema.

***2.- Módulo de Básicos y Productos.***
***Objetivo:*** Dar mantenimiento a los productos y básicos que intervienen en la composición del proyecto.

***3.- Módulo de Partidas.***
***Objetivo:*** Dar mantenimiento a los actividades, partidas, subproyectos y proyectos que se administran en el sistema.

***4.- Módulo de Inventarios.***
***Objetivo:*** Controlar las entradas y salidas, de insumos y herramientas requeridos para llevar a cabo los proyectos.

***5.- Módulo de Subcontratistas.***
***Objetivo:*** Controlar a los subcontratistas y destajistas, que colaboran para la realización del proyecto.

***6.- Módulo de Estimaciones.***
***Objetivo:*** Controlar el avance del proyecto.

***7.- Módulo de Estadísticas.***
***Objetivo:*** Administrar la información estadística del sistema.

***8.- Módulo de Infraestructura.***
***Objetivo:*** Operar la información de tablas, equivalencias, políticas y tarifas necesarias para cálculos, conversiones y normas dentro del sistema.

***9.- Módulo de Guías Contabilizadoras.***
***Objetivo:*** Generar la información contable del sistema de control de proyectos.

***10.- Módulo de Procesos Batch.***
***Objetivo:*** Explotar la información y dotar de procesos diarios, semanales, mensuales y anuales al sistema de control de proyectos.

## 2.4.2 .- Módulo de Seguridad.

En nuestro sistema de control de proyectos, se ha definido un módulo de seguridad que contempla:

- Control de sistemas
- Control de usuarios
- Control de funciones
- Control de claves de acceso

Este módulo es independiente de la seguridad de fábrica que contenga el equipo o el sistema operativo en que se desenvuelva nuestro sistema.

Y es realmente un control para cualquier tipo de acceso a los sistemas que se tengan, ya que no existe un sólo sistema en una organización y con el módulo de seguridad una persona, puede tener diferentes usuarios.

Por ejemplo:

El Director de la empresa tiene necesidad de consultar el sistema de contabilidad, el sistema de cuentas por pagar, el sistema de clientes, el sistema de control de proyectos, así el tendrá un usuario por cada sistema, o un usuario con acceso a cada uno de los sistemas.

Un usuario es una entidad lógica para la operación del sistema.

Llamo función a un número relacionado con una operación bien definida., la cual es realizada por un programa o grupo de programas.

El módulo de seguridad proporciona los medios para controlar los accesos de usuarios al sistema, y administrar los programas y los archivos que componen todo el sistema o los sistemas, ya que debido a la lógica de este módulo, tiene la capacidad de administrar más de un sistema.

Conceptos generales acerca del módulo de seguridad.

El acceso a un sistema de información es algo que debe estar restringido, debido a que la integridad del sistema (programas y datos ) depende de que este modulo funcione correctamente.

Los usuarios del sistema deben tener una clave de acceso y una clasificación como usuarios, es decir, un usuario dentro de un equipo sea del tipo personal o multiusuario, tiene ciertas capacidades y para el manejo y operación del equipo. Por ejemplo un operador tiene la capacidad de operar la consola de control del sistema, montar y desmontar cintas, cartuchos y otros dispositivos de entrada-salida (input-output I/O), el administrador no tiene restricciones, solo las limitantes que determina el mismo equipo, el

usuario del sistema de contabilidad, sólo debe tener acceso a la información de contabilidad el usuario de nomina, solo a la información de nómina; dentro de los mismos sistemas pueden existir clasificaciones de usuarios, que por lo general son los administradores, operadores y usuarios normales.

Las capacidades se limitan a poder consultar la información, consultar y dar de alta información, consultar, dar de alta, dar de baja y cambios.

Cada submódulo del sistema puede ser asignado a un usuario en particular y un usuario en particular, puede tener acceso a todo el sistema.

Supongamos un diagrama de Euler o de conjuntos donde el conjunto universal es el sistema y los elementos son los módulos del sistema. El administrador del sistema o manager system tiene la obligación de administrar todo el sistema y por ello no tiene limitaciones con el mismo. Tiene la capacidad de dar de alta funciones, usuarios, cambios de claves de accesos, cambios de capacidades, y sobre todo la de revisar los archivos de accesos y transacciones que se llevan a cabo por los usuarios del sistema inclusive de el mismo. Los niveles de seguridad dentro de los equipos son independientes de la seguridad que en este módulo será creada. Se tiene la particularidad de crear usuarios genéricos como es el caso de capturistas, consultores, cajeros y otros que sean producto de la necesidad. Por decir un capturista puede tener acceso al módulo de estimaciones, inventarios, infraestructura, destajistas y partidas; pero solo para dar de alta información.

El análista del proyecto tendrá acceso a las mismas funciones, pero con la salvedad de que tiene capacidad de accesar la información, puede de alta, baja y cambios.

El administrador del sistema no tiene limitaciones.

El operador solo tiene posibilidad de efectuar procesos diarios, semanales, mensuales, anuales.

## 2.4.3 .- Módulo de Básicos y Productos.

Este módulo contempla la administración de los básicos, que son por ejemplo el agua, el cemento, la varilla, los clavos, madera, yeso; las herramientas como: pinzas, desarmadores, palas, seguetas; equipo como: revolvedora, grúas, trascabos, camiones, camionetas; dentro de como caso especial también la mano de obra como : albañil, peón, yesero en el caso de un sistema de información operador de computadora, análista de sistemas, Programador de Sistemas Jr., Programador de sistemas Sr., Líder de Proyecto, Asesor.

La diferencia entre un básico y un producto consiste en que el básico es indivisible y el producto es el resultado de la unión o mezcla de básicos, por ejemplo el concreto, es un producto cuyos básicos son: el agua, el cemento, la grava, la arena, la cal, el aditivo, la revolvedora, el porcentaje de mano de obra y el porcentaje de herramientas.

Para la utilización de este sistema ,se ha generado una serie de claves, las cuales son homónimos según sea el caso, a excepción de las claves de proyectos, subproyectos, partidas y conceptos.

En básicos y productos se ha utilizará un homónimo de 8 caracteres, como ejemplo el agua cuya clave tomamos 10AGUA01, los dos primeros caracteres numéricos nos indican que es un básico 10, el 20 será utilizado para las claves de mano de obra es decir 20YESERO que es la clave del Yesero, los 30 son las herramientas y equipos 30PINZ01 son pinzas del número uno.

Dentro del sistema las unidades juegan un papel muy importante, y es necesario manejar magnitudes, que sean propias en el concepto, es decir: si tenemos que las unidades para el concreto más usadas son el metro cúbico, usaremos el metro cúbico, pero para el caso de castillos, por las dimensiones de una casa hogar no es factible utilizar la tonelada para cuantificar el uso de la varilla, por ello utilizaremos el kilogramo, y será necesario hacer la conversión antes de capturar los datos. Ya que se vende por tonelada y no por kilogramo.

Los productos requieren, que al ser dados de alta, se detalle la matriz de composición, y al momento de generar el presupuesto uno de los pasos, será el de explosionar los básicos y grabar el producto.

En el caso de los recursos humanos, se debe tipificar las unidades por horas, jornadas o lapsos de tiempos acordes con las partidas y actividades. Aunque también se pueden utilizar otro tipo de patrones como mejor sea decidido por el análista o responsable del proyecto.

### 2.4.4 .- Módulo de Partidas.

Desde las actividades indivisibles o tareas, y las agrupaciones de estas ( partidas, subproyectos y proyectos) son capturadas y soportadas por este módulo, en el que la información que se maneja es:

- Clave de la actividad, partida, subproyecto y proyecto
- Descripción abreviada
- Descripción detallada
- Unidad o Patrón de medida
- Factor de conversión
- Unidad de medida de conversión
- Precio unitario en moneda nacional
- Factor de conversión
- Clave de la divisa a la que se esta convirtiendo
- Cantidad estimada
- Cantidad real.
- Precio unitario más estimado
- Precio unitario real
- Fecha de inicio más temprano
- Fecha de inicio más tardío
- Fecha de inicio real
- Duración más estimada
- Duración real.
- Clave Responsable.

La manera de operación de este módulo, es la siguiente:

Se da una clave a cada una de las tareas, partidas, subproyectos y proyectos. Esta clave por si misma define la agrupación de las tareas en que partida, el conjunto de partidas en que subproyecto y el conjunto de subproyecto en que proyecto.

A continuación se muestra la codificación, donde:

El primer par de caracteres numéricos de la izquierda define el proyecto, el segundo par de caracteres son los subproyectos, los terceros las partidas, y los últimos la tarea.
El orden de alta debe ser primero el proyecto, seguido el subproyecto, después las partidas y al último las tareas.

Ejemplos:

Al capturar el proyecto 01 la clave quedará 0100000.

Para el subproyecto 05 del proyecto 04 la clave es 04050000.

La partida 88 del subproyecto 45 proyecto 33 quedaría 33458800.

La tarea 77 para el caso anterior es 33458877.

En el caso de un proyecto industrial, al dar de alta una tarea es necesario proporcionar materiales, mano de obra, herramienta y equipo que son utilizados para elaborar dicha tarea, ya que debe relacionarse cada tarea con un producto.

Está información o matriz de datos es proporcionada por un área técnica.

Si se deseará instalar bomba de 1 HP.

La matriz queda de la siguiente manera:

**Materiales:**

1 bomba de 1 HP con accesorios.

**Mano de Obra:**

Un Plomero 4 horas.
Un Supervisor 0.5 hora.

**Herramienta:**

Porcentaje del desgaste de la herramienta, es un porcentaje fijo 5% de la suma de materiales y mano de obra de la tarea.

**Maquinaria y Equipo:**

Nada.

| Código | Cantidad | Precio Unitario | U.M. | Importe |
|---|---|---|---|---|
| 10BOMB10 | 1.0 | 200.00 | PZA | 200.00 |
| 20PLOM01 | 0.5 | 150.00 | JOR | 75.00 |
| 20SUPER1 | 0.0625 | 200.00 | JOR | 12.50 |
| 30PORC01 | 5 | sumatoria | % | 15.00 |
| | | | **TOTAL** | 302.50 |

## 2.4.5 .- Módulo de Inventarios.

El módulo de inventarios administra los básicos, productos, herramientas y mano de obra que intervienen en el proyecto. La manera de controlar lo anterior es mediante claves de entrada, salida, entrada-salida manual y entrada-salida automática. Es simple deducir que lo que entra menos lo que sale es la existencia.

Pero cuanto es posible tener en los almacenes, sin que las pérdidas sean significativas y ¿cuanto es posible no tener en los almacenes, sin que las pérdidas por el retardo en el suministro sean significativas. La pregunta es ¿cuánto dinero en materiales, equipo, maquinaria debo tener antes, ahora y después para un proyecto?, ya que para generar una estimación del costo de un proyecto, son necesarios precios unitarios reales, y si al final del proyecto no se cuanto vale lo que tengo en existencia, no podré saber exactamente cuál fue el costo del proyecto.

Los eventos que se administran en este módulo son:
Las entradas y salidas de básicos, productos, herramientas y la renta de equipos.
Con las entradas se concilia contra las ordenes de pedidos efectuadas automáticamente por los procesos diarios, ya que ello prevé la administración de materiales necesarios para las tareas estimadas en el futuro inmediato, dentro del ciclo de vida del proyecto y de acuerdo con los resultados del avance de obra.

En el caso de un proyecto como el desarrollo de sistemas de información, este módulo es prácticamente minimizado.

### *2.4.5.1.- PRODUCTOS.*

Los productos del módulo de inventarios son:

- Mantenimiento a elementos de inventario.
- Mantenimiento a movimientos de inventario.
- Reportes de entradas, salidas y existencias.
- Catálogo de básicos.
- Catálogo de productos.
- Conciliación de entradas contra pedidos.

## 2.4.6 .- Módulo de Subcontratistas.

Al desarrollarse un proyecto, sobre todo si es grande o debido a la especialización que existe en el mundo contemporáneo, es común utilizar subcontratistas, los colaboradores que aportan alguna actividad de oficio o exclusivamente mano de obra como el caso de yeseros, albañiles, ayudantes, peones y similares, son conocidos como destajistas. El módulo consiste de los siguientes submódulos:

Mantenimiento a subcontratistas y destajistas.
Mantenimiento de estimaciones a subcontratistas y destajistas.
Reporte de estimaciones de subcontratistas y destajistas por rango de fechas.
Reporte para evaluar en obra avances de subcontratistas y destajistas.
Reporte para pagos de subcontratistas y destajistas.
Reporte de estadísticas de subcontratistas y destajistas.
Programa para gráficas de avance de obra de subcontratistas y destajistas.

### *2.4.6.1.- PRODUCTOS*

- Bases de datos actualizadas con la información de subcontratistas y destajistas.
- Contratos de subcontratistas y destajistas.
- Reportes para estimaciones de subcontratistas y destajistas.
- Reportes de avance de obra de subcontratista y destajistas.
- Reportes de pagos a subcontratistas y destajistas.
- Estadísticas de subcontratistas y destajistas.

## 2.4.7 .- Módulo de Estimaciones.

La figura 2.4.6 muestra como en un sistema controlado, la entrada y la salida es analizada y si hay diferencia entre lo que sale y se desea que salga, la entrada es corregida, para que la salida sea la correcta. Este proceso se llama retroalimentación.

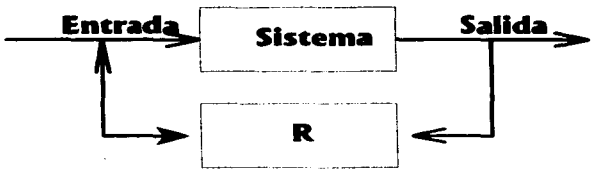



*Figura 2.4.6*

Al integrarse la información de los avances o productos terminados por cada uno de los recursos, se contará con la información mínima necesaria para la toma de decisiones en la retroalimentación.

Un proyecto bien controlado dará como producto final, lo mismo que se previo desde su concepción del mismo; no menos, no más.

### *2.4.7.1.- PRODUCTOS*

- Registro de estimaciones actualizado.
- Reporte de estimaciones por periodo.
- Reporte de estimaciones por recurso.
- Reporte de estimaciones por subproyecto.

## 2.4.8 .- Módulo de Estadísticas.

La estadística aplicada es una de las ramas de las matemáticas que más utilidad a generado a partir de los últimos años. El módulo de estadísticas no plantea como objetivo profundizar en estudios de esta índole dentro de un proyecto, pero si el uso de la información para el control del proyecto. Los programas que lo componen sólo hacen uso de la información generada por los otros módulos y aplican sobre estos técnicas de estadística descriptiva. Un dato valioso es saber la diferencia entre lo estimado contra lo real, tanto de tiempo, como de dinero, y otro punto valioso es poder mostrar gráficamente el presupuesto, o en su caso el grado de avance del proyecto.

### *2.4.8.1.- PRODUCTOS*

- Histogramas de avances estimados vs reales.
- Reportes de estimado vs real.
- Valores promedio en desviaciones.

## 2.4.9 .- Módulo de Infraestructura.

El sistema requiere de una serie de tarifas, catálogos, artículos, tablas, políticas, equivalencias, con las cuales delimitar la operación y soportar ciertas restricciones y rangos de variables de proceso. ejemplos de ello son:

- Factores de conversión de moneda nacional a dólares.
- Fecha del ciclo diario.
- Tablas para el porcentaje de impuesto mensual y anual.
- Tablas para el porcentaje de indirectos.
- Tablas para el rango de fechas por estimación de obra.
- El catálogo de cuentas contables.
- Tablas de equivalencias para el módulo de guías contabilizadoras.
- Tablas para la validación de unidades o patrones de medida.
- Tabla de días festivos y de no laborables.
- Tablas de direccionamiento para personalización del sistema.

Para dar mantenimiento a toda esta información, los submódulos de infraestructura, proporcionan una función en particular para cada una de estas tablas, siendo moldeable a cada aplicación especifica.

Por ejemplo:

La función XXX es para dar mantenimiento a los tipos de cambio de moneda nacional a dólares o cualquiera otra divisa.

La función XXY es para dar mantenimiento al catálogo de cuentas contables.

### *2.4.9.1.- PRODUCTOS*

- Todos aquellos datos necesarios para soporte y operación del sistema.

## 2.4.10 .- Módulo de Guías Contabilizadoras.

## ESPECIFICACIONES DE PROCESOS

### Submódulo de catálogos

Dentro de este submódulo se integra la captura y mantenimiento de los catálogos de conceptos, eventos, equivalencias, cuentas contables para movimientos de control de proyectos; que serán la base para definir las guías contables.

La función de carga del catálogo será una actividad que se realice al iniciar el proceso.

Dentro del mantenimiento de los diferentes catálogos se contará con opciones para cambios, consultas, bajas y reportes.

### Submódulo de guias contables

En este submódulo se diseñarán las guías contabilizadoras utilizando las tablas definidas en el submódulo de catálogos.

El proceso de definición es el siguiente: se definen eventos de operación y conceptos de aplicación para cada elemento de la cuenta contable deseada, y se les asignan valores fijos o variables para ser convertidos posteriormente mediante una tabla de equivalencias. El resultado de esta definición es un conjunto de referencias que al ser resueltas devuelven una cuenta contable del catálogo para cargo y para abono.

Al ser capturada cada guía contable deberá ser probada para determinar que la cuenta resultante sea válida.

### Submódulo contabilizador

En este submódulo se capturarán los movimientos contables que afectan directamente la contabilidad de los movimientos de control de proyectos.

La definición previa de catálogos que a su vez permiten la generación de guías contabilizadoras son indispensables para la aplicación correcta de los movimientos contables.

La contabilización manual permite registrar manualmente los movimientos de control de proyectos o aquellos movimientos extraordinarios que necesiten elaborarse con este procedimiento. Se capturarán los datos que permiten la identificación de la guía contable correspondiente al movimiento en cuestión y una vez verificada la existencia de la cuenta correspondiente se genera el asiento y la póliza contable.

La generación de pólizas contables requerirá de estos elementos importantes:

1. Selección de eventos.
2. Captura de datos e importes por concepto.
3. Generación e impresión de asientos contables.
4. Generación de la póliza contable e integración al histórico.
5. Generación de la póliza contable para la contabilidad general.

**Submódulo de Consultas y Transferencias**

Este submódulo representa la validación final del sistema ya que nos permite revisar la información contable de los movimientos del control de proyectos.

Esta revisión se puede efectuar mediante **consulta** o **transferencias** (consultas no planeadas).

**Consultas.**

Por lo que se refiere a consultas del histórico de movimientos debemos considerar los siguientes conceptos de agrupación y clasificación :

- Fecha de contabilización (puede ser un rango).
- Cuenta contable (puede ser un rango).
- Clave de proyecto (como clave de consumo)

Es importante mencionar que las consultas pueden ser por uno o más de los conceptos antes enunciados por ejemplo:

Los movimientos de una o más cuentas contables de una fecha en particular o de un rango de fechas

Los movimientos existentes para una Clave de proyecto de una cuenta en particular.

O cualquier combinación válida de acuerdo a los conceptos antes mencionados.

Las consultas pueden requerirse en pantalla o papel, y ésta debe ser una opción que puedan manejar los usuarios con libertad, si es posible habrá que prever impresiones en línea.

**Transferencias.**

Este proceso representa el punto medular para la toma de decisiones del usuario del área técnica, aquí se extraerá información para consultarla de manera eficiente y oportuna.

La extracción de movimientos obedecerá a los criterios de selección mencionados anteriormente y será generada en archivos ASCII de manera que puedan ser utilizados para análisis más sofisticados en otras aplicaciones, tales como hojas de cálculo o bases de datos que soporten este formato.

**Submódulo de conciliación**

En determinados puntos en el tiempo será necesario comparar los registros elaborados en el sistema de contabilidad de control de proyectos, cuyo resultado será un listado presentando las diferencias ha ser corregidas en el sistema.

Las opciones de conciliación son:

Comparación de los movimientos registrados en las bases de datos de control de proyectos entre sí, es decir tendremos que comparar el detalle de dos o más cuentas contables de nuestro histórico.

Comparación de los movimientos de una o más cuentas del histórico de control de proyectos contra los movimientos de una o más cuentas del histórico de Contabilidad General.

### *Catálogo de eventos*

Contenido: EVENTOS POSIBLES DE MOVIMIENTOS

Nota: Los registros no podrán borrarse a menos de que no tengan hijos (tabla de conceptos)

| Num | Campo | Descripción | Tipo | L | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | Clave | Clave del evento (Campo único) | C | 2 | 1 | 2 |
| 2 | Nombre | Nombre del evento | C | 40 | 3 | 42 |
| 3 | Abreva | Abreviatura | C | 10 | 43 | 52 |
| | | | | | | |

### *Catálogo de conceptos*

Contenido: CONCEPTOS

| Num | Campo | Descripción | Tipo | L | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | Clave | Clave del concepto ( campo único ) | C | 2 | 1 | 2 |
| 2 | Nombre | Nombre del concepto | C | 40 | 3 | 42 |
| 3 | Abreva | Abreviatura | C | 10 | 43 | 52 |

### *Catalogo de equivalencias*

Contenido: EQUIVALENCIAS DE CONCEPTOS

| Num | Campo | Descripción | Tipo | L | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | Contenido | Clave del contenido (Campo único) | C | 2 | 1 | 2 |
| 2 | Valor | Valor del contenido en este campo podrán ser incorporados elementos que hagan referencia a funciones cuyo resultado represente el valor deseado | C | 20 | 3 | 22 |
| 3 | Equivale | Equivalencia en este campo podrán ser incorporados elementos que hagan referencia a funciones cuyo resultado represente la equivalencia deseada | C | 20 | 23 | 42 |
| | | | | | | |

### *Catálogo de contenidos*

**Contenido:** CONTENIDOS

| Num | Campo | Descripcion | Tipo | Long | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | Clave | Clave del concepto (Campo único) Validar contra el catálogo de conceptos su existencia y validez. | C | 2 | 1 | 2 |
| 2 | Nombre | Nombre del concepto | C | 40 | 3 | 42 |
| 3 | Abrevia | Abreviatura | C | 10 | 43 | 52 |
| 4 | Equivalen | Requiere de equivalencia (Cierto o Falso) | L | 1 | 53 | 53 |
| 4 | Tabla | Archivo de equivalencias (Default Tabla de equivalencias general) | C | 12 | 54 | 65 |

### *Guías contables*

**Contenido:** DEFINICIÓN DE GUÍAS CONTABLES

| Num | Campo | Descripción | Tipo | Longitud | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | Evento | Clave del evento. Validar contra la tabla de eventos | Character | 2 | 1 | 2 |
| 2 | Concepto | Clave del concepto según el evento elegido validar contra la tabla de conceptos la existencia de la llave (Evento + concepto) | Character | 2 | 3 | 4 |
| 3 | Nivel | Nivel de cuenta contable a la que pertenece este (evento+Concepto) el sistema deberá valida que la guía completa incluya todos los niveles posibles de cada cuenta. Por ejemplo: Cuenta de mayor, auxiliar, etc. | Character | 1 | 5 | 5 |
| 4 | Descrip | Descripcion de la guia | Character | 40 | 6 | 45 |
| 5 | Pos_Ini | Posición inicial en la cuenta | Numérico | 2 | 47 | 48 |
| 6 | Ps_Fin | Posicion final en la cuenta | Numérico | 2 | 49 | 50 |
| 7 | Valor fijo | valor reemplazante en la cuenta de ser de longitud Pos_Fin-Pos_ini | Character | 16 | 51 | 66 |
| 8 | Equivalencia | Clave del contenido a evaluarse contra los contenidos o bien contra la tabla de equivalencias | Character | 10 | 67 | 76 |

**Histórico de movimientos contables de control de proyectos.**

Contenido: ***HISTÓRICO DE MOVIMMENTOS CONTABLES DE CONTROL DE PROYECTOS***

| Núm | Campo | Descripción | Tipo | Longitud | P1 | P2 |
|---|---|---|---|---|---|---|
| 1 | CUENTA | Cuenta contable | Character | 4 | 1 | 4 |
| 2 | SCTA | Subcuenta | Character | 4 | 5 | 8 |
| 3 | SSCTA | Subsubcuenta | Character | 4 | 9 | 12 |
| 4 | SSSCTA | Subsubsubcuenta | Character | 4 | 13 | 16 |
| 5 | REFERENC | Referencia hacia la cuenta | Character | 16 | 17 | 32 |
| 6 | DESCRIPC | Descripción del movimiento Centro de costos Proyecto | Character | 40 | 33 | 72 |
| 7 | IMPORTE | Importe del movimiento | Numérico | 12 | 73 | 84 |
| 8 | CVEMOV | Clave de movimiento 1- Abono 2- Cargo | Numerico | 1 | 85 | 85 |
| 9 | MONEDA | Clave de la moneda 1- Moneda Nacional 2.- Dólares | Numérico | 1 | 86 | 87 |
| 10 | POLIZA | Póliza contable | Character | 6 | 88 | 93 |

### *2.4.10.1.- PRODUCTOS*

Los siguientes productos integrarán el módulo de guías contabilizadoras.

**1.- Catálogos de Referencia.**

Mantenimiento de catálogos para la construcción de guías contables
Altas, Bajas, Cambios, Consultas y Reportes

**2.- Guías Contables.**

Mantenimiento de guías contabilizadoras
Altas, Bajas, Cambios, Consultas y Reportes

**3.- Contabilizador.**

Módulo de contabilización que utilizará las guías contables para la correcta captura de datos y generación automática de las pólizas y asientos contables de los eventos a contabilizar ya sea a nivel de detalle o de registro de la contabilidad de mayor.

**4.- Consulta y Transferencia de información de Contabilidad a Control de Proyectos.**

Consultas, reportes y transferencias de información a otras herramientas (archivos ASCII) para responder a consultas no planeadas.

**5.- Conciliación.**

Conciliación de archivo histórico de movimientos de control de proyectos y conciliación contra archivos históricos de contabilidad general.

## 2.4.11 .- Módulo de Procesos Batch.

En los sistemas transaccionales, es parte medular del sistema el ciclo de vida del mismo, ya que el tiempo es un factor apriori.

La valoración de las actividades de un proyecto debe ser periódica, el intervalo de tiempo depende en si misma del proyecto y sus prioridades, como explicación sea el proyecto una intervención quirúrgica, donde el tiempo total de la operación no debe ser mayor a unas cuantas horas, y donde el ciclo de valoración debe ser menor al total de estas, otro caso: la construcción de una planta hidroeléctrica, el tiempo total estimado es de años y donde los puntos de valoración para la retroalimentación pueden ser por día, semana, mes o cualquier periodo o lapso de tiempo menor al total. La relación tiempo de control va ligada a las actividades.

El tiempo en cualquier proyecto es oro y cualquier retraso, puede significar el caos, si no es corregido lo más pronto posible, y para que esto suceda, lo primero es detectarlo.

El módulo de Procesos Batch esta compuesto de procesos automáticos, que son activados por reloj o que bien pueden ser mecanizados, para que un operador en un momento determinado los mande ejecutar.

Previo a los procesos los archivos o base de datos deben ser actualizados con la información generada en el proyecto, previamente vaciada en los reportes de avance de obra, reportes de entradas y salidas de almacenes, reportes de avance de obra de subcontratistas, reportes de altas y bajas de recursos.

Los productos indicarán la fecha y hora de ejecución y el intervalo en el tiempo que cubre cada uno de los reportes.

***2.4.11.1.- PRODUCTOS.***

- Copias en medio magnético de las bases de datos (Respaldos).
- Actualizaciones a las bases de datos.
- Reporte de estimaciones de obra.
- Reporte de existencias en almacén.
- Reporte y ordenes de pedidos para el almacén.
- Reportes múltiples de avances de obra.
- Reporte de avance de obra estimado próximo periodo a evaluar.
- Reporte de movimientos derivados a contabilidad (pólizas contables).
- Conciliación contabilidad vs control de proyectos.
- Conciliación de pagos a subcontratistas vs contabilidad.
- Reportes de eficiencia de los recursos.
- Reportes para pago de subcontratistas.
- Reportes para pago de recursos internos.
- Estadísticas del proyecto.

Ω

# CAPÍTULO 3

## Análisis del Sistema.

**La Excelencia es el arte que se alcanza a través**
**del entrenamiento y el hábito.**
**Nosotros somos los que hacemos repetidamente**
**La Excelencia, entonces, no es un acto aislado sino un hábito.**

**Sócrates.**

## Análisis del Sistema.

### *3.1.- Utilización del Modelos.*

El objetivo primario del análisis es transformar sus dos entradas principales, las políticas del usuario y el esquema del proyecto, en una especificación estructurada. Esto implica modelar el ambiente del usuario con diagramas de flujo de datos, diagrama de entidad-relación y diagramas de transición de estado.

El proceso del análisis, en sus etapas primarias, consiste en desarrollar un modelo ambiental y un modelo de comportamiento para conformar el modelo esencial que representa una descripción formal de lo que el nuevo sistema debe hacer, independientemente de la naturaleza de la tecnología que se use para cubrir los requerimientos.

Dichos modelos se construyen por tres razones:

- Para resaltar las características importantes del sistema y minimizar aquellas menos importantes.
- Para discutir cambios y correcciones a los requerimientos del usuario, a bajo costo y con riesgo mínimo.
- Para verificar que se entiende el ambiente del usuario, y que se ha documentado de tal manera que los diseñadores y programadores puedan construir el sistema.

Lograremos esto al utilizar diagramas gráficos, ya que con ellos se identifican fácilmente los componentes de un sistema y su interfaz. Todos los demás detalles se presentan en documentos textuales de apoyo tales como lo que se conoce *como "especificación del proceso"* y el "*diccionario de datos*".

Asimismo los diagramas gráficos ofrecen y cumplen con una característica primordial: La capacidad de mostrar un sistema por partes en forma descendente. Esto quizás no sea importante para sistemas pequeños, pues de ellos se puede decir todo lo necesario en una o dos páginas, y cualquiera que necesite conocer bien algún aspecto del sistema puede conocerlo en su totalidad. Sin embargo, algunos sistemas como el que nos ocupa, deben subdividirse para un profundo análisis y seguimiento; es como presentar primero el mapa de la República Mexicana, después el de un Estado de interés, posteriormente el de un Municipio, y así seguir hasta el elemento indivisible.

La herramienta gráfica en la que nos vamos a apoyar es el Diagrama de Flujo de Datos.

### *3.2.- Diagrama de Flujo de Datos (DFD)*

El diagrama de flujo de datos es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y "tanques de almacenamiento" de datos. Otras formas de nombrar al diagrama de flujo de datos son las siguientes:

- DFD (Abreviatura que se usará en este trabajo).
- Carta de burbujas.
- Diagrama de burbujas.
- Modelo de proceso.
- Diagrama de flujo de trabajo.
- Modelo de función.

El DFD es una de las herramientas más comúnmente usadas, sobre todo por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja.

Los DFD se utilizaron por primera vez en la ingeniería de software como notación para el estudio del diseño de sistemas. A su vez, la notación se tomó prestada de artículos anteriores sobre teoría de gráficas, y continúa siendo utilizada por los ingenieros de software que trabajan en la implantación directa de modelos de los requerimientos del usuario.

La gran ventaja que presenta el DFD es que prácticamente no requiere explicación; se puede simplemente mirar el diagrama y entenderlo. La notación es sencilla y clara y, en cierto sentido, intuitivamente obvia. Esto es particularmente importante cuando recordamos que quien lo observará no es el analista, sino el usuario.

**Los componentes de un DFD son el *proceso*, el *flujo*, el *archivo* y la *entidad.***

### 3.3.- El Proceso.

El primer componente del DFD se conoce como *proceso*. Los sinónimos comunes son *burbuja*, *función* o *transformación*.

El proceso muestra una parte del sistema que transforma entradas en salidas; es decir, muestra cómo es que una o más entradas se transforman en salidas. El proceso se presenta gráficamente como un círculo o un rectángulo con las esquinas redondeadas, como se muestra en la ***figura 3.1.***


***Figura 3.1*** Representación de un PROCESO.

Se debe destacar que el proceso se nombra o describe con una sola palabra, frase u oración sencilla. En casi todos los DFD el nombre del proceso denota ***lo que hace***.

### 3.4.- El Flujo.

Un *flujo* se representa gráficamente por medio de una flecha que entra o sale de un proceso, tal como se ve en la figura 2.2. Se usa para describir el movimiento de bloques o paquetes de información o sea, un conjunto de datos, de una parte del sistema a otra. Por ello, los flujos representan datos en movimiento, mientras que los archivos representan datos en reposo.

El flujo de la ***figura 3.2*** tiene un nombre, el cual representa el significado del paquete, o sea, un conjunto de datos, que se mueve a lo largo del flujo. Esto quiere decir que el flujo sólo lleva un tipo de paquete, como lo indica su nombre. Es importante considerar que a veces es útil consolidar varios flujos elementales en uno solo, ya que es mejor utilizar un flujo llamado *MUEBLES* en lugar de diversos flujos llamados *SILLA*, *SILLÓN* y *MESA*.

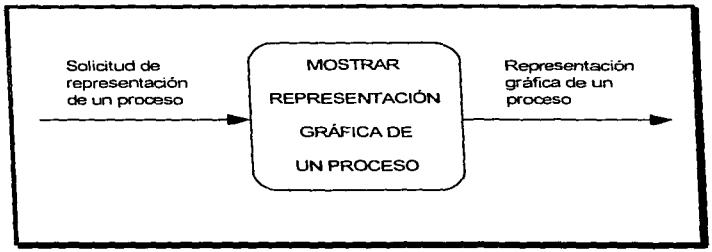



***Figura 3.2*** Flujo de entrada y flujo de salida en un proceso.

Aunque parezca obvio, hay que tener en mente que el mismo contenido pudiera tener distintos significados en distintas partes del sistema. Por ejemplo el número 8724626-3 tiene distinto significado cuando viaja a lo largo de un flujo llamado *NUMERO DE CUENTA* que cuando viaja a lo largo de uno llamado *NÚMERO DE CUENTA VÁLIDO*. En el primer caso, significa un número de cuenta que pudiera ser o no válido; en el segundo caso, significa un número de cuenta que, dentro del contexto del sistema en cuestión, se sabe que es válido.

También hay que notar que las flechas muestran la dirección del flujo, una cabeza de flecha en cualquier extremo (o posiblemente ambos) del flujo indica si los datos se están moviendo hacia adentro o hacia afuera de un proceso (o ambas cosas).

Los flujos de datos pueden divergir o converger en un DFD; conceptualmente esto es algo así como un río principal que se divide en varios más pequeños, o varios pequeños que se unen. Sin embargo, esto tiene un significado especial en un DFD en el cual hay paquetes de datos que se mueven a través del sistema.

En el caso de un flujo divergente, esto significa que se están mandando copias por duplicado de un paquete de datos a diferentes partes del sistema, o bien que un paquete complejo de datos se está dividiendo en varios paquetes individuales más, cada uno de los cuales se está mandando a diferentes partes del sistema. De manera inversa, en el caso de un flujo convergente, significa que varios paquetes elementales de datos se están uniendo para formar agregados más complejos de paquetes de datos.

### 3.5.- El Archivo.

El archivo se utiliza para modelar una colección de paquetes de datos en reposo. En la ***figura 3.3*** se presenta la notación gráfica del archivo.

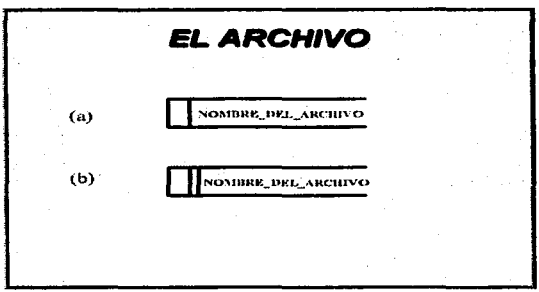



***Figura 3.3***

Representación gráfica del archivo. (a) Notación para dibujarse una sola vez en una hoja. (b) Notación cuando se dibujan dos o más veces en una sola hoja.

De modo característico, el nombre que se utiliza para identificar al archivo es el plural del que se utiliza para los paquetes que entran y salen del archivo por medio de flujos.

Los archivos se conectan por flujos a los procesos. Así, el contexto en el que se muestra un almacén en un DFD es uno de los siguientes:

- Un flujo desde un archivo.
- Un flujo hacia un archivo.

Normalmente se interpreta un flujo que procede de un sistema como una lectura o un acceso a la información del archivo. Esto significa específicamente que:

- Se recupera del archivo un solo paquete de datos; esto es, de hecho, el ejemplo más común del flujo desde un archivo.
- Se ha recuperado más de un paquete del archivo.
- Se tiene una porción de un paquete del archivo.
- Se tienen porciones de más de un paquete del archivo.

Cuando se examinan los flujos que entran y salen de un archivo surgirán muchas preguntas de tipo procedimiento: ¿representa el flujo un solo paquete, muchos, porciones de uno o porciones de diversos paquetes?. En algunos casos, podemos darnos cuenta simplemente viendo la etiqueta del flujo:

Si el flujo no está etiquetado, significa que todo el paquete de información se está recuperando; si la etiqueta del flujo es la misma que la del archivo significa que se recupera todo un paquete (o múltiples instancias de uno completo); si la etiqueta del flujo es diferente del nombre del archivo, entonces se están recuperando uno o más componentes de uno o más paquetes.

Existe un detalle de tipo procedimiento del cual podemos estar seguros, el archivo es pasivo, y los datos no viajarán a lo largo del flujo a menos que el proceso lo solicite explícitamente.

Existe otro detalle de tipo procedimiento que suponen, por convenio, los sistemas de proceso de datos, el archivo no cambia cuando un paquete se mueve del archivo a lo largo del flujo.

Un flujo hacia un almacén habitualmente se describe como una escritura, una actualización o posiblemente una eliminación. Específicamente, sólo puede significar que se tiene una de las situaciones siguientes:

- Se están guardando uno o más paquetes nuevos en el archivo. Dependiendo de la naturaleza del sistema, los paquetes nuevos pudieran anexarse (es decir, de alguna manera acomodarse para que estén "después" de los paquetes existentes); o pudieran colocarse en algún lado entre los paquetes existentes.

Esto es a menudo un asunto de la implantación (es decir, controlado por el sistema específico de administración de bases de datos).

- Uno o más paquetes se están borrando o retirando del archivo.
- Uno o más paquetes se están modificando o cambiando. Esto pudiera traer consigo un cambio de todo un paquete, o (más comúnmente), de sólo una porción, o de una porción de múltiples paquetes.

En todos estos casos es evidente que el archivo cambia como resultado del flujo que ingresa. El proceso (o procesos) conectados con el otro extremo del flujo es el responsable de realizar el cambio al archivo.

## 3.6.- La Entidad Externa o Terminador.

El siguiente componente del DFD es una entidad externa o terminador; gráficamente se representa como un rectángulo, como se muestra en la ***figura 3.4***.

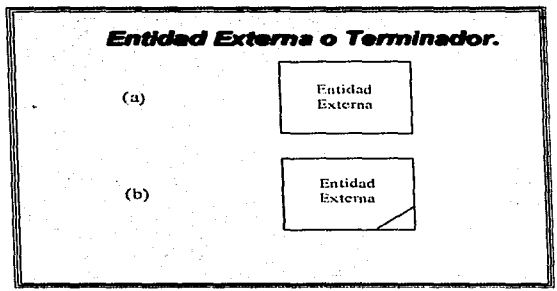


***Figura 3.4***

Representación gráfica de la entidad externa. (a) Notación para dibujarse una sola vez en una hoja. (b) Notación cuando se dibujan dos o más veces en una hoja.

Los terminadores representan entidades externas con las cuales el sistema se comunica. Comúnmente, un terminador es una persona o un grupo, por ejemplo, una organización externa o una agencia gubernamental, o un grupo o departamento que esté dentro de la misma compañía u organización, pero fuera del control del sistema que se está modelando. En algunos casos, un terminador puede ser otro sistema.

Suele ser muy fácil identificar los terminadores en el sistema que se está modelando. A veces el terminador es el usuario; es decir, el usuario suministra datos al sistema o bien los recibe ya procesados. En otros casos, el usuario se considera parte del sistema y ayudará a identificar los terminadores relevantes, es decir, de los que reciben información o bien a los que se la entrega ya procesada.

Existen tres factores importantes que hay que tomar en cuenta en relación a las entidades:

- Son externos al sistema que se está modelando; los flujos que conectan los terminadores a diversos procesos (o archivos) en el sistema representan la interfaz entre él y el mundo externo.
- Como consecuencia, es evidente que ni el analista ni el diseñador del sistema están en posibilidad de cambiar los contenidos de un terminador o la manera en que trabaja.
- Las relaciones que existan entre los terminadores no se muestran en el modelo de DFD. Si existen relaciones entre los terminadores y si es esencial modelarlos para poder documentar los requerimientos del sistema, entonces, por definición, los terminadores son en realidad parte del sistema y debieran modelarse como procesos.

## 3.7.- Guía para la construcción de un DFD.

Existen reglas adicionales que se requieren para poder utilizar los DFD con éxito. Algunas de estas reglas ayudarán para no elaborar DFD erróneos, incompletos o lógicamente inconsistentes. Y tienen la finalidad de auxiliar en el dibujo de un DFD grato a la vista, y que, por tanto, tenga más probabilidades de que lo lea con cuidado el usuario

1 Identificar las entradas y salidas netas.

- Determinar el contexto del sistema, esto es las entidades externas e internas al sistema.
- Buscar los flujos de datos que cruzan el contexto, estos constituirán las entradas y salidas netas.
- El contexto debe ser suficientemente grande como para incluir todo lo relevante al esfuerzo de desarrollo, pero también lo suficientemente pequeño para no incluir cuestiones irrelevantes.

2 Llenar el cuerpo del DFD.

- Concentrarse en los flujos de datos.
- Averiguar si dentro de los procesos identificados hay flujos de datos, en tal caso separar ese proceso en dos, tres o en los procesos que sean necesarios.
- Cuestionarse en cada flujo de datos:
  + ¿Qué se necesita para construirlo?
  + ¿De dónde provienen sus componentes?
  + ¿Qué procesos se requieren para ello?
  + ¿Podemos construir este flujo de datos a partir de los flujos de datos de entrada?

3 Introducir en el DFD los archivos que representen los depósitos de información que el usuario ha identificado.

4 Revisar si no hay flujos de datos faltantes o sobrantes o redes desconectadas.

5 Nombrar los flujos de datos.

- Asignar nombres a todos los flujos de datos.
- Los flujos que entran o salen de un archivo no requieren nombre en la mayoría de los casos.
- Evitar nombres ambiguos.
- No usar homónimos o sinónimos al nombrar un flujo de datos.
- El nombre del flujo de datos no únicamente debe representar los datos sino también lo que sabemos acerca de ellos.
- Tener cuidado de no mezclar datos que nada tienen que ver entre sí.
- Si no se encuentra un nombre preciso, dividir el flujo de datos.

6 Nombrar los procesos.

- Asegurarse de que todos los flujos de datos ya tienen nombre.
- Tratar de nombrar los procesos usando un verbo y un objeto directo.

Un proceso en un DFD puede representar una función que se está llevando a cabo, o pudiera indicar cómo se está llevando a cabo, identificando a la persona, grupo o mecanismo involucrado. En el último caso, obviamente es importante etiquetar con precisión el proceso, de modo que quienes leen el DFD, especialmente los usuarios, puedan confirmar que se trata de un modelo preciso. Sin embargo, si el proceso lo hace una sola persona, se recomienda identificar el papel que dicha persona está representando, más que su nombre o identidad.

- Los nombres de los procesos deben hacerse en términos de sus entradas y salidas.
- Evitar ambigüedades o nombres que abarquen parcialmente el proceso.
- Si se obtienen nombres ambiguos reparticionar el proceso.
- No utilizar verbos muy generales como "Procesar", "Hacer"; ni objetos directos muy generales como "Información", "Datos", etc.

7 Asignar un número a los procesos.

- Como una forma conveniente de referirse a los procesos en un DFD se asigna un numero a cada burbuja. No importa mucho cómo se haga esto, de izquierda a derecha, de arriba a abajo o de cualquier otra manera servirá, mientras haya consistencia en la forma de aplicar los números. La única situación que hay que tener en cuenta es que el sistema de numeración implicará, para algunos lectores casuales de un DFD, una cierta secuencia de ejecución. Esto no es así en absoluto. El modelo de DFD es una red de procesos asíncronos que se intercomunican, lo cual es, de hecho, una representación precisa de la manera en la que en realidad muchos sistemas operan.
- La utilidad de la numeración se aprecia también cuando un proceso necesita dividirse en otros procesos más elementales -nombrados "hijos"-, ya que éstos heredan el número del proceso "padre". Por ejemplo, a un proceso X se le asigna el número 2, otros procesos derivados de éste tendrán los números 2.1, 2.2, 2.3, etc. Los números de los procesos derivados del 2.1, serán 2.1.1, 2.1.2, etc.

8 Omitir detalles de manejos de errores.

- Cuando se detecten rutas de error se recomienda dejarlas indicadas y trabajar en ellas posteriormente.
- Si el error no requiere descartar algún proceso ya efectuado, ignorarlo.
- Si el error requiere deshacer algo que ya se hizo entonces no ignorarlo.

9 Mostrar flujo de datos y no de control.

- Verificar que en todos los flujos de datos se esté pasando información, de otra manera, eliminarlos.
- Verificar que la información en cada flujo de datos esté usándose en el proceso al que llega, de no ser así, eliminarla.

10 Revisión del DFD.

- Es imposible que el DFD esté correcto la primera vez.
- Si el primer DFD dibujado resulta ser el final, seguramente tiene errores.
- Ejecutar todo este proceso de manera iterativa.

Esto pudiera parecer mucho trabajo, pero bien vale la pena el esfuerzo de desarrollar un modelo preciso, congruente y agradable, de los requerimientos del sistema.

- Estar preparado para sustituir el primer DFD por otras versiones mejoradas.
- Este proceso iterativo dará grandes beneficios en las fases posteriores del ciclo de vida del sistema.

Ahora bien, existen otras reglas respecto a cómo asegurar que el DFD mismo sea consistente, estas son:

- Evitar procesos infinitos, es decir, aquellos que tienen entradas pero no salidas. Estos procesos también son conocidos como agujeros negros.
- Evitar los procesos de generación espontánea, que tienen salidas sin tener entradas, por que son sumamente sospechosas y generalmente incorrectas.
- Estudiar profundamente los flujos y procesos sin nombre, ya que esto indica que puede ser divido en datos o procesos más elementales.
- Analizar cuidadosamente los archivos de "sólo lectura" o "sólo escritura".

Un archivo típico debiera tener tanto entradas como salidas. La única excepción a esta regla es algún archivo externo, que sirve de interfaz entre el sistema y alguna entidad externa.

Cuando el sistema a modelar es intrínsecamente complejo y tiene docenas o incluso cientos de procedimientos que modelar, se recurre a la construcción de un DFD global en una serie de niveles de modo que cada uno proporcione sucesivamente más detalles sobre una porción del nivel anterior.

El DFD del primer nivel consta sólo de un proceso, que representa el sistema completo; los flujos de datos muestran las interfaces entre el sistema y las entidades externas junto con los archivos externos que pudiera haber. Este DFD especial se conoce como Diagrama de Contexto.

El DFD que sigue del diagrama de contexto se conoce como el DFD 0. Representa la vista de más alto nivel de las principales funciones del sistema, al igual que sus principales interfaces.

Algunos aspectos que se deben considerar adicionalmente son:

- Saber cuántos niveles debe haber en un DFD. Cada DFD debe tener no más de media docena de burbujas y archivos relacionados. Así, si se ha partido un sistema grande en tres niveles, pero sus DFD de nivel más bajo aún contienen 50 procesos cada uno, entonces falta por lo menos un nivel más.

Si no podemos escribir una especificación razonable para un proceso en alrededor de una página, entonces es probable que sea demasiado compleja y debiera partirse en DFD de menor nivel antes de tratar de escribir la especificación.

- No existen reglas acerca del número de niveles en el análisis de un sistema típico. En un sistema simple, probablemente se encontrarán dos o tres niveles; uno mediano tendrá generalmente de tres a seis niveles; uno grande tendrá de cinco a ocho niveles. El número total de procesos se incrementará exponencialmente a medida que se baja de nivel al inmediato inferior.

- No es necesario dividir todas las partes del sistema con el mismo nivel de detalle. Algunas partes del sistema pueden ser más complejas que otras y pueden requerir uno o más niveles de partición.

- Mostrar los niveles al usuario. Muchos usuarios sólo querrán ver un diagrama: un usuario ejecutivo de alto nivel pudiera querer ver tan sólo el diagrama de contexto o tal vez el diagrama de nivel 0; un usuario de nivel operacional pudiera querer ver tan sólo la figura que describe el área en la cual está interesado. Pero si alguien quiere ver una parte más extensa del sistema, o tal vez todo, entonces tiene sentido presentar los diagramas de una manera descendente: comenzar con el diagrama de contexto y continuar hasta niveles más bajos de detalle.

- Asegurar que los niveles del DFD sean consistentes entre sí. Para asegurar que cada figura sea consistente con su figura de más alto nivel se sigue una regla sencilla: los flujos de datos que salen y entran de un proceso en un nivel dado deben corresponder con los que entran y salen de toda la figura en el nivel inmediato inferior que la describe.

- Mostrar los archivos en los diversos niveles. Esta es una área donde la redundancia se introduce deliberadamente en el modelo. La regla es la siguiente: mostrar un archivo en el nivel más alto donde primeramente sirve de interfaz entre dos o más procesos; luego, mostrarlo de nuevo en CADA diagrama de nivel inferior que describa más a fondo dichos procesos de interface. Otra consideración importante es que los archivos locales, que utilizan sólo los procesos en una figura de menor nivel, no se mostrarán en niveles superiores, dado que se incluyen de manera implícita en un proceso del nivel inmediato superior.

- Realizar la partición de los DFD en niveles. A pesar de que los DFD deben presentarse al público usuario de una manera descendente, no es necesariamente cierto que se deban desarrollar así. El enfoque descendente intuitivamente es muy atractivo, sin embargo, un enfoque que tiene más éxito es identificar primero los acontecimientos externos a los cuales debe responder el sistema y utilizarlos para crear un primer borrador del DFD.

### *3.8.- El Diccionario de Datos.*

El diccionario de datos es otra herramienta del modelado del sistema; si bien es cierto que no tiene el atractivo gráfico de los DFD, es indispensable.

Sin el diccionario de datos, el modelo de los requerimientos del usuario no puede considerarse completo; todo lo que se tendría en un borrador rudimentario, una "visión del artista" del sistema.

La importancia del diccionario de datos a menudo les pasa de largo a muchas personas, pues no han utilizado algún tipo de diccionario durante 10 ó 20 años. Es seguro que sin él, llegará el momento en que el usuario lector se extraviará y no podrá estar seguro de que entendió los detalles de la aplicación.

El diccionario de datos es un listado organizado de todos los datos pertinentes al sistema, con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento común de todas las entradas, salidas, componentes de archivos y cálculos intermedios. El diccionario de datos define los datos haciendo lo siguiente:

- Describe el *significado* de los flujos y archivos que se muestran en los DFD.
- Describe la *composición* de agregados de paquetes de datos que se mueven a lo largo de los flujos, es decir, paquetes complejos (por ejemplo el domicilio de un cliente), que pueden descomponerse en unidades más elementales (como ciudad, estado y código postal).
- Describen la *composición* de los paquetes de datos en los almacenes.
- Especifica los *valores* y *unidades* relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.
- Describe los detalles de las relaciones entre los archivos que se enfatizan en un diagrama de entidad-relación.

### 3.9.- La necesidad de la Notación en el Diccionario de Datos.

En la mayoría de los sistemas reales con los que se trabaja, los conjuntos o elementos de datos, serán lo suficientemente complejos como para que se necesite describirlos en términos de otras cosas. Los elementos complejos de datos de definen en términos de elementos más sencillos, y los sencillos en términos de los valores y unidades legítimos que pueden asumir.

**Notación del Diccionario de datos.**

Existen muchos esquemas de notación comunes utilizados por el analista de sistemas. El que se muestra en la ***figura 3.5*** es de los más comunes y utiliza varios símbolos sencillos.

| | |
|---|---|
| = | está compuesto de |
| + | y |
| ( ) | optativo (puede estar presente o ausente) |
| { } | iteración |
| [ ] | seleccionar una de varias alternativas |
| * *comentario* * | comentario o descripción |
| # | identificador (campo clave) para un archivo |
| \| | separa opciones alternativas en la construcción |

***Figura 3.5*** Notación del diccionario de datos.

Como puede apreciarse, los símbolos parecen algo matemáticos y pudiera pensarse que serían difíciles de entender. Sin embargo, como veremos pronto, la notación es bastante fácil de leer.

## 3.10.- Definiciones.

La definición de un dato se introduce con el símbolo "=". En este contexto, el "=" se lee: "se define como", o "se compone de", o simplemente "significa".

Por ejemplo la notación Z = X + Y se puede leer de las siguientes formas:

Z esta descompuesto de X y Y.

o Z se define como X y Y.

Para definir por completo un dato, se debe incluir lo siguiente:

- El significado del dato dentro del contexto de la aplicación del usuario. Por lo común se utiliza la notación: "* *significado o comentario* *".
- La composición del dato, si se compone de partes elementales con significado.
- Los valores que puede tomar el dato, si se compone de partes elementales con significado.
- Los valores que puede tomar el dato, si es un dato elemental que no puede descomponerse más.

**Elementos de Datos Básicos.**

Las partes elementales de los datos son aquellas para las cuales ya no existe una descomposición con significado dentro del contexto del ambiente del usuario. Esto usualmente es una cuestión de aplicación y es algo que se debe explorar cuidadosamente con el usuario. Pero tal vez en algunos ambientes de usuario no se requiere tal descomposición, ni sea relevante, ni tenga significado.

Cuando se han identificado los datos elementales, deben introducirse al diccionario de datos. Como se indicó anteriormente, el diccionario de datos debe proporcionar una breve narrativa, encerrada entre caracteres "*", que describa el significado del término en el contexto del usuario. Desde luego, habrá términos que se definan solos, es decir, cuyo significado es universal para todos los sistemas de información, o donde el analista pudiera estar de acuerdo en que no se necesita aclarar más.

Por ejemplo apellido paterno, peso actual, sexo, teléfono particular, etc.

Para tales casos se usa la notación "**" para indicar "sin comentarios", ya que no se escribe algún texto explicando el significado del dato. Sin embargo, es importante especificar los valores y unidades de medida que pueden tomar.

**Datos Opcionales ( )**

Un dato opcional, como la frase implica, es aquel que puede estar o no presente en un dato compuesto.

Por ejemplo salón de clases = (ubicación) + (nombre del salón) significa que un salón puede ser identificado sólo por su ubicación o bien, sólo por su nombre o bien, por su ubicación y nombre o aunque no es usual por ninguno de los dos.

**Iteración { }**

La notación de iteración se usa para indicar la ocurrencia repetida de un componente de un dato. Se lee como "cero o más ocurrencias de".

Por ejemplo inscripción = nombre del alumno + domicilio del alumno + {materia} significa que la inscripción siempre debe contener un nombre de alumno, un domicilio y también cero o más ocurrencias de una materia. Es decir, es un alumno que está inscrito en una, o dos o más materias.

Cuando se quieren especificar límites inferior y/o superior se anotan en los extremos de las llaves; como se indica aquí 1{materia}5, un alumno se puede inscribir en 1, 2, 3, 4 ó 5 materias.

**Selección [ ]**

La notación de selección indica que un dato consiste en exactamente un elemento de entre un conjunto de opciones alternativas. Las opciones se encierran en corchetes "[" y "]", y se separan por una barra vertical "|".

Por ejemplo sexo = [Femenino | Masculino]

**Alias**

Un alias, como el término implica, es una alternativa de nombre para un dato. Esto es una ocurrencia común cuando se trata con diversos grupos de usuarios en diferentes departamentos o ubicaciones geográficas, que consisten en utilizar distintos nombres para decir lo mismo. El alias se incluye en el diccionario de datos para que esté completo, y se relaciona con el nombre primario u oficial del dato.

Por ejemplo: investigador = * alias de profesor *

En este caso, la composición de investigador no se anota debido a que es un alias de profesor y de esta manera "hereda" la composición de éste último.

***3.11.- Implantación del Diccionario de Datos.***

En un sistema mediano o grande, el diccionario de datos puede requerir una cantidad considerable de trabajo y tiempo.

Lo más recomendable es hacer uso de la computadora para introducir definiciones al diccionario, verificar que estén completas y consistentes, y producir reportes apropiados.

Se deben tener presentes, además, las siguientes limitaciones posibles:

- Estar forzado a limitar los nombres de datos a cierta longitud. Esto probablemente no sea un gran problema, pero podría el usuario insistir en un nombre como ***nombre-completo-alumno*** y que el paquete de elaboración del diccionario obligue a abreviarlo como ***nom-com-alu***.
- Pudiera ser que el usuario no permita ciertos caracteres al elaborar el nombre del dato, como el guión ( - ) por ejemplo.
- El diccionario de datos siempre debe estar completo y terminado antes de llevarse a cabo la elaboración del seudocódigo y obviamente del código de los programas. Para lograrlo quizá sea necesario asignar dos o más analistas para su elaboración.

Ω

# CAPÍTULO 4

# Diseño del Sistema.

**"El amor es el medio, la verdad es el fin, si utilizamos el medio, tarde o temprano llegaremos al fin, a la verdad, a Dios".**
**Mahatma Gandhi.**
**(1869-1948)**

**La gente viaja para maravillarse**
**ante las cumbres de las montañas,**
**ante las olas enormes de los mares,**
**ante los grandes cauces de los ríos,**
**ante la vasta extensión del océano,ante**
**el imponente movimiento de los astros y ...**
**pasan ante ellos mismos sin maravillarse.**

**San Agustín.**

# CAPITULO 4

## Diseño del Sistema.

El diseño es el primer paso para desarrollar cualquier producto o sistema de ingeniería. El diseño lo podemos definir como: " El proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física ".

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

El diseño del software se asienta en el núcleo técnico del proceso de ingeniería de software y se aplica independientemente del modelo de desarrollo de software seleccionado. Una vez que se han establecido los requerimientos del software, el diseño de software es la primera de tres actividades técnicas: diseño, generación del código (construcción) y prueba, cada paso transforma la información de forma que finalmente se obtiene un software de computadora validado.

El flujo de información durante las tres etapas técnicas de la creación del producto de software se muestra en la ***figura 4.1***. Los requisitos del software, establecido mediante la relación de requerimientos, el modelo de flujo de datos, y el modelo de entidad-relación ayudan y alimentan a la etapa de diseño. En la etapa de diseño se realiza el diseño de datos, el diseño arquitectónico y el diseño procedimental.

El diseño de datos transforma el modelo del campo de información, creado durante el análisis, en el diccionario y las estructuras de datos que se van a requerir para implementar el software. El diseño arquitectónico define la estructura del producto de software y la interface con el usuario. El diseño procedimental transforma los elementos estructurales en una descripción procedimental del software.

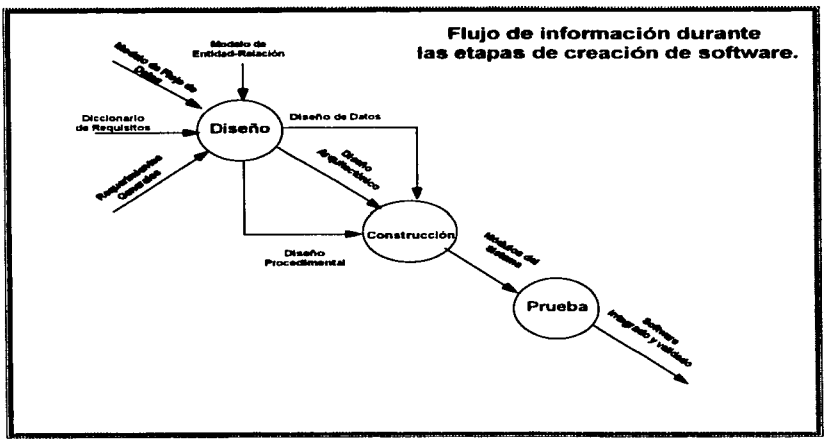



**Figura 4.1.** Flujo de información durante las etapas de creación de software.

## Diagrama del Proceso de Diseño Lógico.

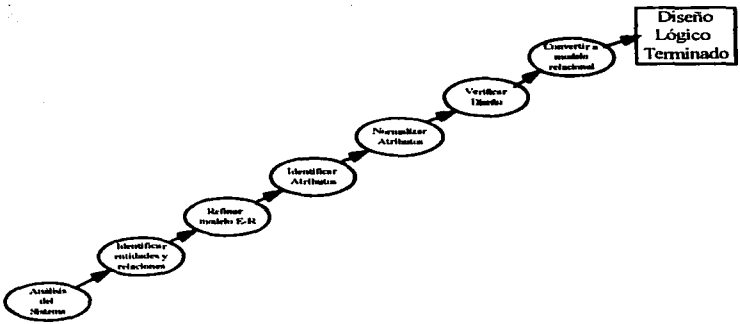



***Figura 4.2.***

Diagrama del proceso de diseño lógico.

### *4.1.- Modelo Conceptual de Datos.*

El modelo conceptual de datos es el primer paso del proceso TOP-DOWN para el desarrollo de una base de datos, se realiza en las etapas de Análisis y Diseño del Ciclo de Desarrollo de un Sistema. Representa una forma efectiva para integrar y documentar los requerimientos de información de una organización, así como independiente del hardware o del software usados para la implementación.

El objetivo del Modelo Conceptual de Datos es desarrollar el Modelo Entidad-Relacion (E-R).

Un modelo (E-R) puede ser utilizado para una base de datos de red, jerárquica o relacional.

### *4.2.- Componentes del Modelo Entidad-Relación.*

#### 4.2.1.- Entidades.

Son los aspectos importantes acerca de los cuales se necesita conocer información y a los que se les pueden asignar atributos.

Para dibujar una entidad se utilizan cuadros o rectángulos de cualquier dimensión con las esquinas redondeadas. El nombre de la entidad se escribe en mayúsculas y en singular. En ocasiones también se escribe un nombre alternativo o sinónimo, el cual se anota abajo del principal y entre paréntesis. Finalmente, el nombre de los atributos en minúsculas como se muestra en la figura 4.3.

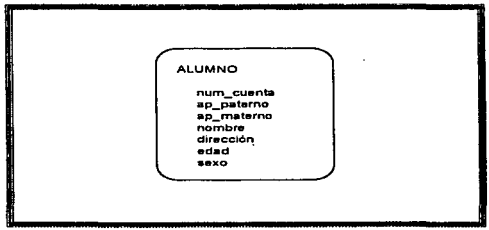



***Figura 4.3***

Ejemplo de cómo dibujar una entidad.

Se debe estar seguro de que la entidad posea múltiples "ocurrencias", "instancias" o "registros", considerando que cada instancia de la entidad tiene valores específicos para cada atributo de la entidad.

Cada instancia debe ser identificada como única de otras instancias de la misma entidad. Un atributo o conjunto de atributos que identifican de manera única a una instancia dentro de una entidad son llamados **Identificadores Únicos (UID)**.

***Si una entidad no puede tener un identificador único (UID), ésta no puede ser una entidad.***

Los atributos que identifican de manera única una entidad y pertenecen a los UID de las entidades son precedidos por un símbolo de número y asterisco **( # * )**.

### *4.2.2.- Relaciones.*

Evento bidireccional que representa la asociación entre dos entidades, o entre una entidad consigo misma.

La forma de escribir una relación debe contemplar el formato que se presenta en la ***figura 4.4***:

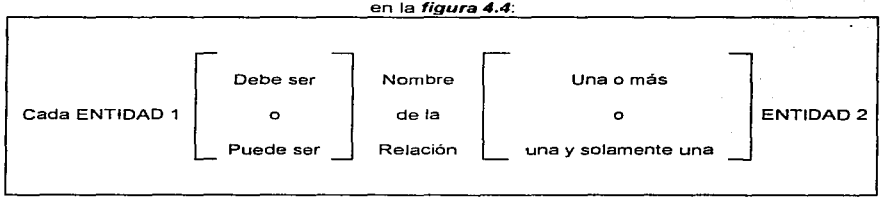



***Figura 4.4***

Formato para escribir la relación entre dos entidades.

Por ejemplo, si tuviéramos las entidades PROFESOR y MATERIA, su relación estaría dada como sigue:

- Cada materia puede ser enseñada por uno y solamente un profesor.
- Cada profesor puede ser asignado a una o más materias.

Es importante destacar que cada dirección de una relación tiene:

- Un nombre. Por ejemplo: Enseñado *por* o asignado *a*.
- Una opción. Por ejemplo: *Debe* ser o *puede* ser.
- Un grado o cardinalidad. Por ejemplo: *Uno y solamente uno*, o *uno* o *más*.

Un grado de 0 es etiquetado como *puede ser.*

Para dibujar la relación se necesita una línea entre dos entidades y los nombres de las relaciones escritas en minúsculas.

La opcionalidad queda esquematizada en la ***figura 4.5***

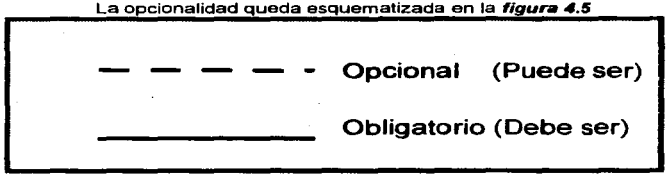



***Figura 4.5***
Opcionalidad.

El Grado queda ilustrado en la ***figura 4.6***

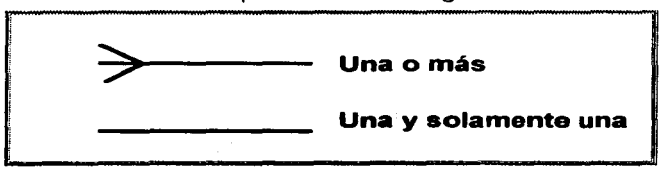



***Figura 4.6***
El grado representado por una "pata de gallo" y una línea simple.

En resumen, lo anterior se ejemplifica en la ***figura 4.7:***

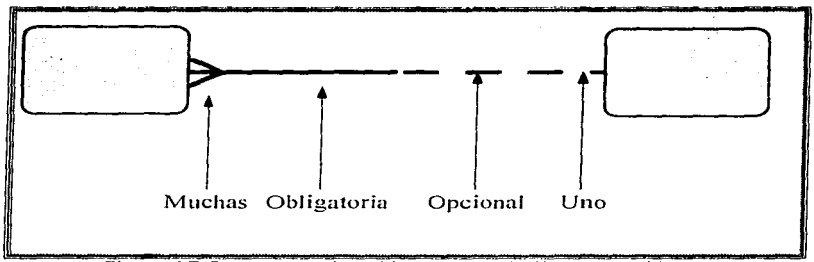



***Figura 4.7*** Representación gráfica de una relación entre entidades

**4.2.3.- Grados de Relación.**

Existen tres grados de relación:

- Relaciones muchos a uno.
- Relaciones muchos a muchos.
- Relaciones uno a uno.

Una relación uno a uno (1 a 1 ó I:I) tiene un grado de *uno y solamente* uno en ambas direcciones.

Una relación muchos a uno (M a 1 ó M:1) tiene el grado de *uno o más* en una dirección y el grado de *uno y solamente uno* en la otra dirección.

Una relación muchos a muchos (M a M o M:M) tiene el grado de *uno o más* en ambas direcciones. Usualmente son opcionales en ambas direcciones, aunque puede ser opcional también en una sola dirección.

**4.2.4.- Matriz de Relaciones.**

Se usa una Matriz de Relaciones como una ayuda para la recolección inicial de información sobre las relaciones entre una serie de entidades.

Estándares de Matriz de Relaciones.

- Esta matriz muestra si están relacionadas y en qué forma cada entidad (renglón) con cada entidad (columna) .

- Todas las entidades están dispuestas en el lado izquierdo y en la parte superior de la matriz.

- Si una entidad está relacionada con otra entidad, entonces el nombre de esta relación se muestra en la celda de intersección.

- Si una entidad no está relacionada con otra entidad, entonces se dibuja una línea en la celda de intersección.

- Cada relación por encima de la diagonal es el inverso o imagen espejo de la relación por debajo de la línea diagonal.

- Las relaciones recursivas (o sea, una entidad relacionada consigo misma) son representadas por las celdas en la diagonal.

En la ***figura 4.8*** se representa una matriz de relaciones.

| | Entidad 1 | Entidad 2 | . . . | Entidad n |
|---|---|---|---|---|
| Entidad 1 | ———— | Relación 1-2 | . . . | Relación 1-n |
| Entidad 2 | Relación 2-1 | ———— | . . . | ———— |
| . . . | . . . | . . . | . . . | . . . |
| Entidad n | Relación n-1 | ———— | . . . | ———— |

***Figura 4.8***

***Matriz de relaciones.***

**4.2.5.- Representación del Diagrama Entidad-Relación (E-R)**

El diagrama E-R debe realizarse de tal manera que sea fácil de leer por la gente que necesita trabajar con él. Sus características son:

Limpio y Ordenado.

- Alinear las cajas de las entidades.
- Dibujar las líneas de relación como rectas horizontales o verticales.
- Usar un ángulo de 30° a 60° el cual facilita seguir las líneas de la relación cuando éstas se cruzan.
- Evitar el uso de muchas líneas paralelas, ya que dificulta el seguimiento.

Texto Claro.

- Escribir el texto con claridad, en forma concisa.
- Evitar abreviaciones y modismos.
- Agregar adjetivos para mejorar el entendimiento.
- Alinear el texto horizontalmente.
- Poner el nombre de la relación al final de la línea y en lados opuestos de la línea.

Formar Fáciles de Recordar.

- Hacer el Diagrama E-R fácil de recordar. Que la gente recuerde las formas y los patrones.
- No se deben dibujar diagramas E-R en una cuadrícula.
- Compactar en la medida de lo posible las cajas de las entidades para ayudar a la visualización del diagrama.

Reglas de Formato.

- Tratar de posicionar las "patas de gallo" como las de las figuras 4.6 y 4.7, en la parte izquierda para líneas horizontales y en la parte superior para las líneas verticales.
- Posicionar las entidades más volátiles arriba y a la izquierda del diagrama.
- Posicionar las entidades menos volátiles abajo y a la derecha del diagrama.

**4.2.6.- Atributos.**

Información específica que se necesita conocer o tener acerca de una entidad. Los atributos describen una entidad para calificar, identificar, clasificar, cuantificar o expresar el estado de una entidad. Dicho de otra manera, representan un tipo de descripción o detalle, mas no una instancia.

Los atributos deben tener nombres claros para el usuario y no codificados para el desarrollador. Un atributo debe estar asignado en una sola entidad y deben ser descompuestos hasta su mínimo componente.

También es importante verificar que un atributo no sea derivado o calculado de los valores existentes de otros atributos.

***4.2.6.1.- Opcionalidad de Atributos.***

***Atributos obligatorios.***

Un valor *debe ser* conocido por cada ocurrencia de la entidad.
Marcarlo con un asterisco (*).

***Atributos opcionales.***

Un valor *puede ser* conocido por cada ocurrencia de la entidad.
Marcarlo con una o minúscula.

### 4.2.6.2.- *Identificador de Atributos.*

Identificar atributos examinando las notas de entrevistas y realizando preguntas al usuario. Los atributos pueden aparecer en notas de entrevistas como:

- Frases y palabras descriptivas.
- Sustantivos.
- Frases preposicionales.
- Pronombres y sustantivos posesivos.

Asignar identificadores únicos.

Un identificador único (UID) es cualquier combinación de atributos y/o relaciones que sirven para identificar en forma única una ocurrencia o instancia de una entidad. Cada ocurrencia de una entidad debe ser identificada de manera única.

En algunas ocasiones el identificador único de una entidad forma parte del identificador único de otra, en estos casos de utiliza la *barra UID*, como en la ***figura 4.9***

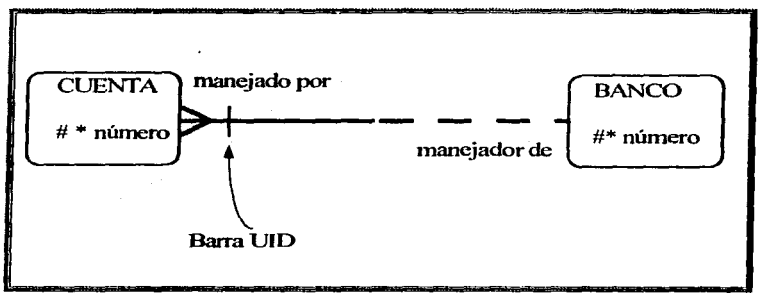



***Figura 4.9***

La barra UID indica que la relación con el BANCO es parte del UID de CUENTA.

Una relación incluida dentro de un UID debe ser obligatoria y *una y solamente una* en la dirección que participa en el UID.

Una entidad puede ser identificada de manera única a través de múltiples relaciones como lo indica la ***figura 4.10.***

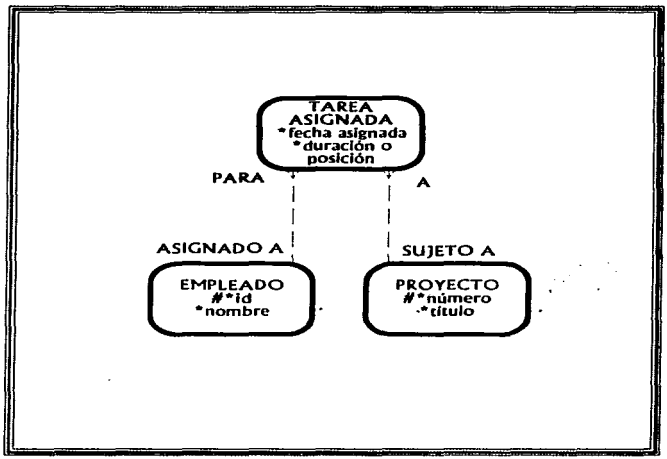



***Figura 4.10.a***

La TAREA ASIGNADA no tiene UID.

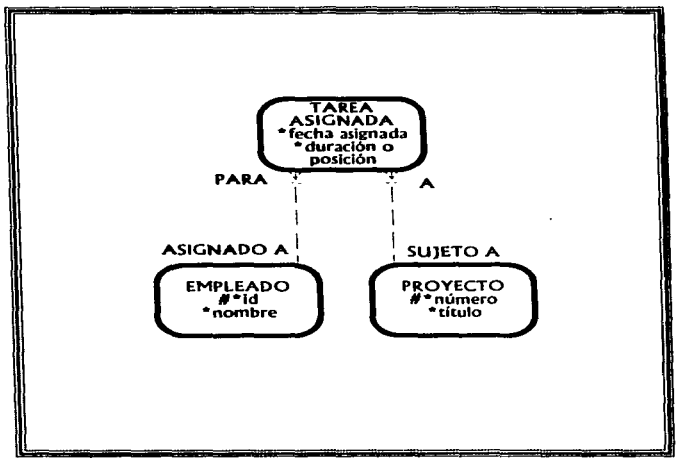



***Figura 4.10 b.***

Una TAREA ASIGNADA es identificada de manera única por el EMPLEADO a quien se le dio la TAREA ASIGNADA, por el PROYECTO que contiene la TAREA ASIGNADA y por la fecha asignada.

Ambas relaciones son obligatorias y una y sólo una, en la dirección incluida en el UID.

Una entidad puede tener más de un UID como se ilustra en la ***figura 4.11***

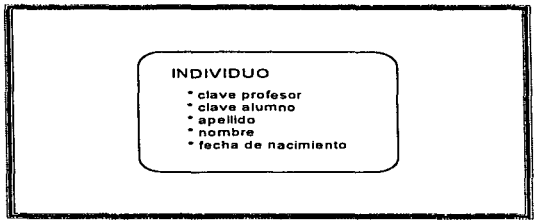



***Figura 4.11***

Candidatos a UID: "clave profesor", "clave alumno" y "apellido/nombre".

Aquí probablemente la combinación de apellido y nombre no es única.

En casos como el anterior se elige un candidato UID para ser el UID primario, y los otros para ser los UID secundarios tal como lo muestra la figura 4.12.

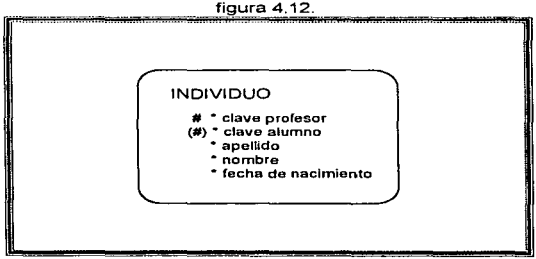



***Figura 4.12.***

Cada UID se marca con (#) o se deja sin ninguna marca.

Cuando una entidad no ofrezca posibles candidatos para ser UID, es válido crear uno artificial que asuma dicho papel. Esto se ilustra en la ***figura 4.13***.

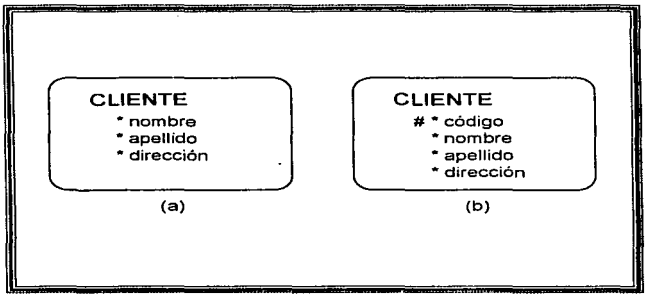



***Figura 4.13.***

En la entidad CLIENTE (a) no se aprecia un posible UID. Por lo que fue necesario crear un atributo artificial "código" en la entidad CLIENTE (b) para habilitarlo como UID.

## 4.3.- Modelo Conceptual de Datos Avanzado.

### 4.3.1.- Normalizar el Modelo de Datos.

Normalizar es un concepto de bases de datos relacional, pero sus principios se aplican al Modelo Conceptual de Datos.

Para validar cada atributo se deben utilizar las reglas de normalización que se muestran en la ***figura 4.14.***

| Regla de la Forma Normal | Descripción |
|---|---|
| Primera Forma Normal (1FN) | Todo los atributos deben tener un sólo valor para cada instancia. |
| Segunda Forma Normal (2FN) | Un atributo debe ser dependiente del identificador único completo. |
| Tercera Forma Normal (3FN) | Ningún atributo no-UID puede ser dependiente de otro atributo no-UID. |

***Figura 4.14***

Reglas de normalización.

Un modelo de datos entidad-relación normalizado se traslada automáticamente dentro de un diseño de base de datos.

La tercera forma normal es un objetivo generalmente aceptado para eliminar redundancia en un diseño de base de datos.

Las formas normales arriba de la tercera no son comúnmente utilizadas.

*4.3.1.1.- Regla de la Primera Forma Normal.*

Todos los atributos deben tener un sólo valor para cada instancia.

VALIDACIÓN

Validar que cada atributo tenga un valor único para cada ocurrencia de la entidad. Ningún atributo deberá tener valores repetidos.

Si un atributo tiene múltiples valores, se crea una entidad adicional y lo relaciona con la entidad original mediante una relación M:1

*4.3.1.2.- Regla de la Segunda Forma Normal.*

Un atributo debe ser dependiente del identificador único completo.

**Validación.**

Validar que cada atributo dependa completamente del UID. Cada instancia especifica del UID debe determinar una sola instancia de cada atributo.

Validar que un atributo no dependa de una sola parte del UID de la entidad.

Si un atributo no es dependiente del UID completo, está fuera de lugar y deberá ser movido.

### 4.3.1.3.- *Regla de la Tercera Forma Normal.*

Ningún atributo no-UID puede ser dependiente de otro atributo no-UID

**Validación.**

Validar que cada atributo no-UID no dependa de otro atributo no UID.

Mover cualquiera atributo no-UID que dependa de otro atributo no-UID.

Si un atributo depende de otro atributo no-UID, es necesario mover ambos, el atributo dependiente y el atributo del que depende, a una nueva entidad relacionada con la entidad actual.

Cuando algunos atributos se asocian con relaciones M:M (muchos a muchos) se tienen que resolver al sustituir una entidad intersección y dos relaciones M:1 (muchos a uno).

Por ejemplo, si consideramos la relación M:M entre PRODUCTO y VENDEDOR, ilustrada en la ***figura 4.15*** ¿Cuál es el precio actual de un PRODUCTO específico para un VENDEDOR específico?

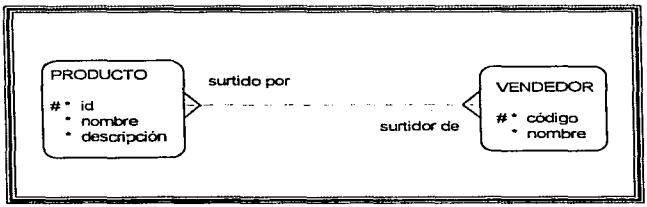



***Figura 4.15***

Ejemplo de una relación M:M

El precio actual parece ser un atributo de la relación entre PRODUCTO y VENDEDOR.

Los atributos únicamente describen entidades. Si los atributos describen relaciones, las relaciones deberán ser resueltas.

La relación M:M entre PRODUCTO y VENDEDOR puede ser resuelta agregando una entidad intersección llamada CATÁLOGO. El precio actual es un atributo de la entidad CATÁLOGO, tal como se muestra en la ***figura 4.16.***

Una vez que está definida la entidad CATÁLOGO, se analizan los requerimientos del cliente para ver la posibilidad de agregar atributos como: cantidad del paquete y unidad de medida. El UID del CATÁLOGO esta compuesto por las dos relaciones.

Una entidad Intersección es frecuentemente identificada por las dos relaciones que le dieron origen, como se ve en la ***figura 4.16***

Las relaciones desde una entidad intersección son siempre obligatorias.

Las entidades intersección son muy comunes para representar situaciones de negocios en el mundo real.

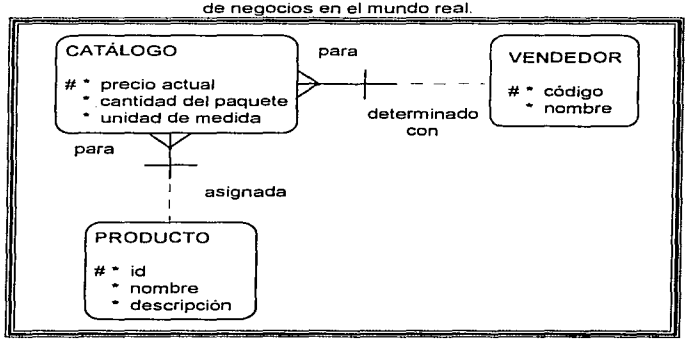



***Figura 4.16***
Ejemplo de una Entidad Intersección.

Las entidades intersección normalmente generan requerimientos para atributos adicionales como el uso de cantidades y fechas. Estas tienden a ser entidades volátiles y de volumen alto.

No olvidar que el UID de una entidad intersección está frecuentemente compuesta por las relaciones entre las entidades que le dieron origen.

Si se resuelven las relaciones M:M al final de la fase de Análisis se puede llegar a tener una entidad Intersección sin atributos.

Una entidad intersección sin atributos es justamente una lista de referencia cruzada en doble sentido entre las ocurrencias de las entidades.

Una entidad intersección sin atributos es la excepción a la regla de que una entidad debe tener atributos para ser una entidad.

### *4.4.- Otros Modelos.*

Existen otros tipos de modelos para representar las relaciones entre entidades, como los que a continuación se presentan:

#### 4.4.1.- Modelo Jerárquico de Datos.

Representación jerárquica de datos como un conjunto de relaciones muchos a uno.

#### 4.4.2.- Modelo Recursivo.

En el modelo recursivo se representan relaciones entre una entidad y ella misma.

#### 4.4.3.- Modelo de Roles.

Este modelo se usa cuando una misma persona puede estar considerada en varias entidades.

#### 4.4.4.- Modelo de Subtipos.

Usar subtipos para modelar exclusivamente tipos de entidad que tienen atributos o relaciones comunes.

Un supertipo es una entidad que tiene subtipos y puede ser separado en dos o más subtipos mutuamente excluyentes. También puede tener atributos y relaciones compartidas por sus subtipos.

#### 4.4.5.- Modelos de Relaciones Exclusivos.

Para modelar dos o más relaciones mutuamente excluyentes de la misma entidad se usa un arco.

Las relaciones en arco frecuentemente tienen los mismos nombres de relación.

Las relaciones de arco deben ser o todas obligatorias o todas opcionales.

Un arco pertenece a una sola entidad y solamente debe incluir relaciones originadas de esta entidad.

Una entidad puede tener múltiples arcos, pero una relación específica solamente puede participar en un sólo arco.

#### 4.4.6.- Modelos de Datos de Tiempo.

Se utilizan cuando se requiere agregar entidades y relaciones al modelo Entidad-Relación para acomodar datos históricos.

Una entidad intersección se usa frecuentemente para mantener información acerca de las relaciones que cambian con el tiempo.

## 4.5.- Conceptos de Base de Datos Relacional.

Una Base de Datos Relacional es una base de datos que es percibida por el usuario como una colección de relaciones o de tablas de dos dimensiones.

Las tablas de Base de Datos Relacional son sencillas pero disciplinadas.

Una Base de Datos Relacional debe tener Integridad de Datos, sus datos deben ser precisos y consistentes.

El Instituto Nacional Americano de Estándares (ANSI) ha establecido a SQL como el lenguaje estándar para operar sobre las Bases de Datos Relaciónales.

Una Base de Datos Relacional puede soportar un conjunto completo de operaciones relaciónales. Las operaciones relaciónales manipulan conjuntos de valores de datos. Las tablas pueden ser utilizadas en la creación de otras tablas. Las operaciones relaciónales pueden ser anidadas.

### 4.5.1.- Llaves Primarias.

Una Llave Primaria (PK) es una columna o grupo de columnas que identifican de manera única a cada renglón en una tabla. Cada tabla debe tener una llave primaria y una llave primaria debe ser única, no pueden cambiar de valor. El UID de una entidad irá de acuerdo con la Llave Primaria en su tabla correspondiente.

Una Llave Primaria que consta de múltiples columnas se llama Llave Primaria Compuesta. Las columna de una Llave Primaria Compuesta deben ser únicas en combinación. Las columnas pueden tener duplicados en forma individual, pero en combinación, no se permiten duplicados.

Ninguna parte de la Llave Primaria puede ser NULA.

Una tabla puede tener más de una columna o combinación de columnas que pueden servir como la Llave Primaria de la tabla. Cada una de estas es llamada Llave Candidata o Alterna.

Todas las Llaves Alternas deben ser únicas y NO NULAS. Los UID secundarios concuerdan con las Llaves Alternas. Los nombres de personas normalmente no son Llaves Alternas por que no se puede garantizar que sean únicas.

#### 4.5.2.- Llaves Foráneas.

Una Llave foránea (FK) es una columna o combinación de columnas en una tabla, que se refieren a una llave primaria en la misma o en otra tabla.

Las llaves foráneas son utilizadas para hacer "JOIN" entre tablas, se basan en los valores de los datos, son puramente lógicas, pueden ser repetidas, nulas y pueden coincidir con un valor de una Llave Primaria existente.

Si una llave foránea es parte de una Llave Primaria, la FK no puede ser NULA.

#### 4.5.3.- Integridad de Datos.

La integridad de datos se refiere a la exactitud y consistencia de los datos.

#### 4.5.4.- Constraints de Integridad de Datos.

Los constraints de integridad de datos definen el estado relacional correcto de la base de datos, aseguran que los usuarios realicen únicamente operaciones en las cuales dejarán a la base de datos en un estado correcto y consistente.

Todos los constraints de integridad de datos deben ser forzados por el DBMS o el software de aplicación.

Un dato es inconsistente si existen múltiples copias de un registro y no todas las copias han sido actualizadas. Una base de datos inconsistente puede proveer información incorrecta o contradictoria a los usuarios.

Los requerimientos del usuario también pueden determinar el estado correcto de una base de datos. Estos requerimientos son llamados Constraints de Integridad de Datos definidas por el usuario.

Los constraints de los datos definidos por el usuario pueden ser administrados por políticas o ser requeridos por las leyes gubernamentales.

Frecuentemente esos requerimientos son completamente arbitrarios o al menos parecen ser arbitrarios.

Los constraints de integridad de datos definidos por el usuario pueden incluir múltiples columnas y tablas.

## 4.6.- Diseño de la Base de Datos.

El diseño de la base de datos es ejecutado durante la etapa de diseño del ciclo de desarrollo del sistema y es ejecutado conjuntamente con el diseño de aplicaciones.

El diseño de base de datos se lleva a cabo por medio de las siguientes actividades:

1. Pasar el modelo E-R a tablas relaciónales para producir el diseño inicial.
2. Refinar el diseño inicial para producir un diseño completo de la base de datos.

La etapa de diseño de la base de datos produce especificaciones de diseño para una base de datos relacional, incluyendo definiciones para tablas relaciónales, índices, vistas y espacio de almacenamiento.

En este punto es necesario documentar cada tabla relacional en un mapa de instancias como el que se muestra en la ***figura 4.17***.

| | columna 1 | columna 2 | columna 3 | . . . | columna n |
|---|---|---|---|---|---|
| Tipo Llave | | | | | |
| Nulos/Único | | | | | |
| Ejemplo | | | | | |
| | | | | | |
| | | | | | |

***Figura 4.17*** Mapa de instancias.

Los tipos válidos de llaves son PK para una columna llave primaria y FK para la columna llave foránea.

Se usan sufijos para distinguir entre múltiples columnas FK en una tabla, por ejemplo, FK1 y FK2. Etiquetar múltiples columnas con el mismo sufijo

Se usa NN para una columna que debe ser definida como NO NULA.

Se usa U para la columna que debe ser única.

Si múltiples columnas deben ser únicas en combinación, etiquetarlas con un sufijo, por ejemplo U1.

Se etiqueta una columna sencilla PKO como NN, U.

Se etiquetan múltiples columnas PK (compuestas) como NN, U1 o como NN, U1, U.

#### 4.6.1.- Diseño Inicial de la Base de Datos.

Para producir el diseño inicial de la base de datos se llevan a cabo los siguientes pasos:

1. Mapear las entidades para las tablas.
2. Mapear atributos para columnas y documentar datos simples.
3. Mapear identificadores únicos a llaves primarias.
4. Mapear relaciones a llaves foráneas.
5. Elegir opciones de arco (si las hay).
6. Elegir opciones de subtipo (si las hay).

#### *4.6.1.1.- Mapear las Entidades.*

Mapear la tabla para cada entidad implica crear un mapa de instancias para la nueva tabla. Por ejemplo, para una entidad llamada EJEMPLO se creará una tabla llamada EJEMPLO.

El nombre de la tabla debe ser fácil de identificar con el nombre de la entidad. El nombre en plural de una entidad se usa algunas veces porque la tabla debe contener un grupo de renglones.

Una entidad simple no es un subtipo o supertipo.

#### *4.6.1.2.- Mapear Atributos a Columnas.*

Se mapea cada atributo de la entidad a una columna en su tabla correspondiente. Se establecen los atributos obligatorios para columnas NO NULAS (NN). El nombre de los atributos deben ser cortos pero significativos.

El nombre de las columnas debe ser fácil de identificar en un modelo E-R.

Es importante prevenir al usuario de no usar las palabras reservadas de SQL para nombres de columnas.

Usar abreviaciones consistentes que no causen confusión al usuario y al programador.

Los nombres de columnas cortos o pequeños reducirán el tiempo requerido parta el comando de SQL "parsing".

#### *4.6.1.3.- Mapear UIDS a Llaves Primarias.*

Asignar cualquier atributo(s) que sea parte del UID de la entidad a columnas PK, las cuales deben etiquetarse.

Todas las columnas etiquetadas con PK deben etiquetarse también con NN y U.

También se asigna un UID que incluya múltiples atributos a una PK compuesta y se etiquetan estas columnas NN y U1.

Si una entidad incluye una relación, agregar columnas de llaves foráneas para la tabla y señalarlas como parte de la llave primaria.

Se debe escoger un nombre único para cada columna FK, y etiquetar la(s) columna(s) PK, NN y FK.

Si existen múltiples columnas FK en una tabla, usar sufijos para distinguirlos, por ejemplo, FK1 y FK2. Etiquetar múltiples columnas llave con el mismo sufijo.

Las PK compuestas deben ser únicas en combinación y deben ser etiquetadas como U1.

Siempre se deben agregar ejemplos de datos para las columnas FK.

#### *4.6.1.4.- Mapear Relaciones Para Llaves Foráneas.*

Para una relación de entidades M:1 tomar el PK de la tabla (1) y ponerlo en la tabla (M).

Se elige un nombre único para la columna FK y etiquetar la(s) columna(s) FK.

Para las relaciones debe ser , se etiqueta la columna como NN y se agregan ejemplos.

Si el PK de la tabla incluye una llave foránea (FK), las columnas FK que soportan la relación, pueden ser agregadas en el tercer paso.

Para una relación obligatoria 1:1, se colocan las FK como únicas en el mapa de instancias en donde la relación es obligatoria y se usa el constraint NO NULO para forzar la condición obligatoria.

Si una relación 1:1 es opcional en ambas direcciones mover la FK en la tabla en cualquiera de las dos tablas de la relación.

La FK para la relación 1:1 debe ser siempre única, pero permite NULOS.

Para una relación recursiva 1:M se agrega una columna FK a la tabla. Esta columna FK debe referenciar valores de la columna PK.

La columna FK hace referencia a un renglón en la misma tabla.

El nombre de la columna FK debe reflejar la relación.

Una FK recursiva nunca debe ser NO NULA.

Para una relación recursiva 1:1 se agrega una FK única a la tabla. Esta columna FK debe referenciar un valor de la columna PK.

La combinación de columnas PK y FK siempre debe ser única para asegurar la relación 1:1. Poniendo el PK y FK como únicos (U) se garantiza que la combinación será única.

Una FK recursiva nunca debe ser NO NULA.

*4.6.1.5.- Escoger Opciones de Arco.*

Los Arcos representan un tipo de llave foránea de alternativa múltiple. Se debe escoger entre las alternativas de diseño de Arco explicito y de Arco genérico para "mapear" arcos a llaves foráneas.

También se utilizan estos dos tipos de diseño para implementar múltiples llaves foráneas cuando un arco atraviesa un conjunto de relaciones 1:1.

Los arcos solamente pueden atravesar el final de las relaciones que son o todas obligatorias o todas opcionales.

El diseño de Arco Explicito crea una columna de llave foránea para cada relación que incluye el arco. También este diseño soportará llaves foráneas múltiples con diferentes formatos. El software de aplicación debe forzar a una relación de exclusividad entre las llaves foráneas.

El diseño de Arco Genérico crea una columna de llave foránea única y una columna para marcar la relación del arco. Como las relaciones son exclusivas, solamente un valor de FK existirá para cada renglón en la tabla.

Si las relaciones debajo del arco son obligatorias, se hacen ambas columnas NO NULAS (NOT NULL). Las llaves foráneas deben compartir el mismo formato para todas las tablas referenciadas. La relación de exclusividad se forza automáticamente.

*4.6.1.6.- Escoger Opciones de Subtipos.*

Se debe escoger entre las alternativas de diseño de una sola tabla, diseño de tablas separadas e implementación del Arco para "mapear" subtipos a tablas.

No olvidar que los subtipos heredan todos los atributos del supertipo y sus relaciones, que los subtipos pueden tener atributos y relaciones propias y, finalmente, que los subtipos deben ser mutuamente excluyentes.

En relación al ***diseño de subtipo en una sola tabla***, se crea una sola tabla para el supertipo y se mapean los subtipos dentro de esta tabla. La tabla sencilla contendrá instancias de todos los subtipos.

Los pasos para llevar a cabo este diseño son:

1. Crear una sola tabla para el supertipo.
2. Crear una columna para cada uno de los atributos del supertipo.
3. Crear una columna TIPO para identificar a que subtipo pertenece cada renglón.
4. Crear una columna para cada uno de los atributos del subtipo.
5. Crear columnas FK para cada una de las relaciones del supertipo.
6. Crear columnas FK para cada una de las relaciones del subtipo.

El diseño de subtipo de una sola tabla requiere una nueva columna *tipo* para identificar el subtipo al que pertenece cada renglón.

Se debe usar un diseño de subtipo en una sola tabla cuando hay pocas relaciones y atributos propios. Las ventajas que se tienen son que el acceso al supertipo es directo y que el subtipo puede ser accesado y modificado usando vistas. Pero las desventajas son que los requerimientos del subtipo NO NULO no se pueden forzar a nivel de base de datos y que la lógica de las aplicaciones tendrán que manejar diferentes conjuntos de atributos, dependiendo del TIPO.

En cuanto al ***diseño de subtipo en tablas separadas*** se debe mapear una tabla para cada subtipo ya que cada una de ellas contendrá solamente instancias de un subtipo.

Los pasos para realizar este tipo de diseño son:

1. Crear una tabla para cada subtipo.
2. En cada tabla subtipo, crear columnas para los atributos del subtipo.
3. En cada tabla subtipo, crear columnas para los atributos del supertipo.
4. En cada tabla subtipo, crear columnas FK para las relaciones del subtipo.
5. En cada tabla subtipo, crear columnas FK para las relaciones del supertipo.

Se usa un diseño de subtipo de tablas separadas cuando hay muchas relaciones y atributos de subtipos específicos.

Las ventajas de este diseño son que la opcionalidad de los atributos se forza a nivel de la base de datos y la lógica de las aplicaciones no requiere de revisión para los subtipos.

Las desventajas son que la consulta al supertipo requiere de un operador de UNIÓN, las vistas que enlazan dos tablas (mediante un join), son solamente de consulta, el código del programa de aplicación debe ser específico para cada tabla individual y el mantenimiento de los UID´s de los subtipos es difícil de implementar.

**4.6.2.- Normalización de Tablas.**

La normalización minimiza la redundancia de los datos. Un dato sin normalizar es redundante.

La redundancia de datos causa problemas de integridad. Las transacciones de actualización y borrado pueden no ser consistentes en todas las copias de los datos causando inconsistencia en la base de datos.

La normalización ayuda a identificar entidades, relaciones y tablas mal diseñadas.

Cada valor de una llave no primaria DEBE depender únicamente de la llave completa, solo de ella y no de ningún otro campo.

Las reglas de las formas normales se muestran en la ***figura 4.18.***

| Regla de la Forma Normal | Descripción |
|---|---|
| **Primera Forma Normal (1FN)** | **La tabla debe tener un sólo valor para cada renglón. La tabla no puede contener grupos repetidos** |
| **Segunda Forma Normal (2FN)** | **La tabla debe estar en 1FN. Cada columna que no es llave debe ser dependiente de la llave primaria completa.** |
| **Tercera Forma Normal (3FN)** | **La tabla debe estar en 2FN. Una columna que no es llave primaria no debe depender de otra columna llave.** |

***Figura 4.18***

Reglas de las Formas Normales.

La tercera forma normal es un objetivo normalmente aceptado para un diseño de base de datos para eliminar la redundancia.

Las formas normales posteriores ya no son utilizadas.

#### 4.6.2.1.- *Reconocer Datos sin Normalizar.*

Un dato sin normalizar no cumple con ninguna regla de normalización.

#### 4.6.2.2.- *Conversión a la Primera Forma Normal.*

Se deben remover los grupos de repetición o vectores de la base de datos y crear una nueva tabla con la PK de la tabla base y el grupo de repetición.

#### 4.6.2.3.- *Conversión a la Segunda Forma Normal.*

Se deben determinar cuales columnas, que no son llave, no dependen de la llave primaria completa de la tabla. También se deben remover esas columnas de la tabla base y crear una segunda tabla con esas columnas y la(s) columna(s) de la PK de la cual dependen.

Si cada columna no depende de la llave primaria completa, la tabla no está en 2FN.

Cualquiera tabla con una llave primaria de una sola columna está automáticamente en la 2FN.

*4.6.2.4.- Conversión a la Tercera Forma Normal.*

Se deben determinar que columnas son dependientes de otra columna no llave. Después se remueven esas columnas de la tabla base y al final se crea una segunda tabla con esas columnas no llave de la cual son dependientes.

Una tabla está en Tercera Forma Normal si una columna no llave no es funcionalmente dependiente de otra columna no llave.

Una columna no llave no puede ser funcionalmente dependiente de otra columna no llave.

*4.6.2.5.- Especificar la Integridad Referencial.*

El valor de una columna de llave foránea debe coincidir con un valor existente en otra columna que sea llave primaria (o ser NULO). Usar los constraints de la integridad referencial para especificar como se debe mantener la integridad referencial.

Se recomienda especificar el Constraint de borrado para definir qué debe suceder si un renglón que contiene una llave primaria referenciada es borrada, así como especificar un Constraint de actualización para definir qué debería suceder cuando una llave primaria es actualizada.

*4.6.2.6.- Diseño de Índices.*

Un índice es asociado con una sola tabla física y contiene los valores de una o más columnas de esa tabla.

Al usar índices se mejora considerablemente el acceso a datos.

Los índices proveen acceso rápido a los renglones de datos y evitan búsquedas en toda la tabla.

Facilitan uniones (join) entre tablas.

Aseguran que no existe un valor duplicado si se define como único.

Son usados automáticamente cuando se referencia en las cláusulas WHERE de las instrucciones de SQL (siempre y cuando la columna no sea modificada).

Un índice concatenado es un índice creado en un grupo de columnas en una sola tabla.

También se usan índices para implementar llaves y soportar los requerimientos de acceso de la aplicación.

Los índices únicos se crean para llaves primarias.

Generalmente los índices no únicos para llaves foráneas.

La creación de índices también se considera para llaves alternativas (índices únicos), en columnas no llaves críticas usadas en las cláusulas WHERE y para cualquier llave de búsqueda.

Los índices requieren espacio y generan una sobre carga al momento de hacer actualizaciones.

Un índice único hace referencia a una columna o grupo de columnas que tienen valores únicos en la tabla.

Índices no únicos hacen referencia a columnas o grupos de columnas que no son únicas en la tabla.

Tener en cuenta que bajo ciertas condiciones, los índices no son usados por el **RDBMS**.

Ω

# CAPÍTULO 5

## Desarrollo del Sistema.

**Bástale a cada día su propio afán.**
**¿Por qué, te preocupas de tantas cosas?**
**¿Por qué, llevas el peso de un ayer que**
**lamentas, si ya no está en tus manos?**
**¿Por qué, te angustia el temor de un**
**mañana que quizá no va haber?**
**El ayer pasó, el mañana no ha llegado...**
**llena bien el hoy que tienes en tus manos,**
**aprovéchalo: ¡¡llénalo!!**
**Piensa que el hoy es tu día para luchar,**
**para vencer, para reparar, para amar.**

San Mateo VI-34.

**Podré no estar de acuerdo con lo que dices,**
**pero daría la vida, por defender tu derecho a decirlo...**

Voltaire

Político Francés ( 1694 - 1778 )

# CAPITULO 5

# Desarrollo del Sistema.

Al paso de crear la base de datos y la codificación de los programas, es llamado el desarrollo del sistema. Un proceso que transforma el diseño en un lenguaje de programación y que da la pauta para la división lógica que integran la construcción de la base de datos y la programación del sistema.

Antes de construir una base de datos del sistema, es necesario conocer algunos conceptos básicos para tener una mejor idea de que es lo que se realizo y como se llevo a cabo, a continuación se mencionan los conceptos necesarios para entender la construcción de la base de datos.

## 5.1 Implementación de la base de datos.

Un sistema de base datos es un sistema de mantenimiento de registros basados en computadora, es decir, un sistema cuyo propósito general es registrar y mantener información. En la figura 5.1 se muestra una representación de un sistema de base de datos Data Base Management System (DBMS). Los cuatro componentes principales son: datos, hardware, software y usuarios.

Datos: Los datos almacenados en el sistema se dividen en una o más bases de datos. Aunque se puede pensar desde el punto de vista didáctico que sólo hay una base de datos, la cual contiene todos los datos almacenados en el sistema. Una base de datos es un repositorio de datos almacenados y en genoral es integra y compartida. es integra por que la base de datos puede considerarse como una unificación de varios archivos independientes donde se elimina parcial o totalmente cualquier redundancia entre los mismos. Y es compartida porque, partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos y utilizarla con propósitos diferentes.

Hardware: El hardware se compone de los volúmenes de almacenamiento secundario (discos duros, tambores magnéticos), donde reside la base de datos, junto con dispositivos asociados como las unidades de control.

Software: Entre la base de datos física en si (es decir, el almacenamiento real de los datos) y los usuarios del sistema existe un nivel del software, que a menudo recibe el nombre de sistema de administración de base de datos o DBMS.

Usuarios: Se considera aquí tres clases de usuarios: 1) programador de aplicación, es el encargado de escribir los programas de aplicación que utilice la base de datos; 2) usuarios finales, es el que recurre a un programa de aplicación escrito por

un usuario programador que acepte órdenes desde la terminal y a su vez formule solicitudes al DBMS en nombre del usuario final; 3) administrador de la base de datos, es el encargado de crear, mantener y administrar la base de datos.

Los usuarios de un DBMS son de tres clases: 1) programador de aplicación, que es el encargado de escribir los programas de aplicación que utilice la base de datos; 2) Usuarios finales, es el que recurre a un programa de aplicación escrito por un usuario programador que acepte ordenes desde la terminal y a su vez formule solicitudes al DBMS en nombre del usuario final; 3) Administrador de la base de datos, es el encargado de crear, mantener y administrar la base de datos, es decir, controlar el acceso, controlar el espacio en disco utilizado, y sobre todo garantizar la integridad de la base de datos y el buen funcionamiento del DBMS.

Así; un sistema de base de datos es un sistema computarizado de información para el manejo de datos por medio de paquetes de software llamados sistemas de manejo de base de datos. Los componentes principales de un sistema de base de datos son el software DBMS y los datos por manejar. Las ventajas de utilizar un DBMS es al menos en cierta medida reducir la inconsistencia y la redundancia, compartir datos, poder hacer cumplir las normas establecidas, aplicar restricciones de seguridad y conservar la integridad.

La construcción de la base de datos del sistema depende entonces de las características propias del DBMS y del modelo de datos seleccionado para el sistema de estimación y control de proyectos.

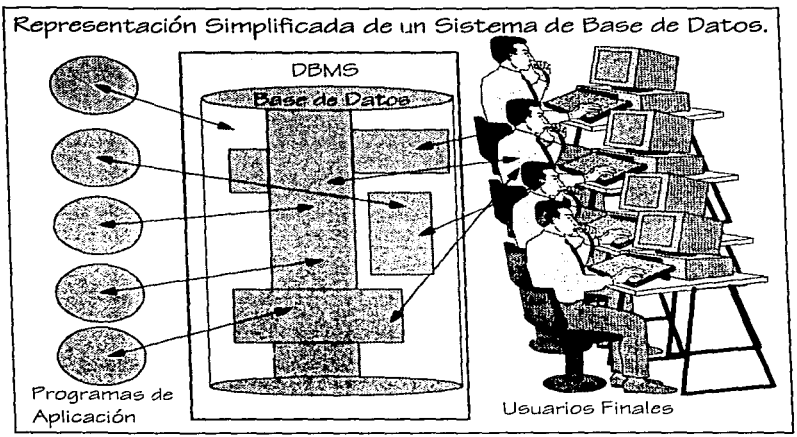



***Figura 5.1 Representación simplificada de un Sistema de Base de Datos.***

**5.1.1. Sistema de Manejo de Bases de Datos (DBMS).**

Un DBMS es un paquete de programación que corre en el sistema operativo central como un largo y sofisticado programa de aplicación. Regularmente el DBMS depende del sistema operativo central. Se clasifican en tres categorías: red, jerárquica y relacional. Esta clasificación es según la estructura lógica a nivel externo, es decir como el usuario percibe la estructura de la base de datos que será mapeada por el DBMS hacia su almacenamiento físico.

El DBMS proporciona al usuario herramientas para la descripción de datos por ejemplo el Data Description Lenguaje DDL, para describir esquemas conceptuales y externos, el DML Data Manipulation Lenguage para manejar los datos de la base, desde su concepción el pasar del diseño a la creación del esquema y subesquema, dicta todo lo concerniente a la base de datos inclusive también el modo de sus operaciones de entrada/salida.

En la practica un buen diseñador puede utilizar un producto, ya predefinido como estándar en una organización de manera aceptable. Aunque si es posible seleccionar lo mas conveniente es no definir la adquisición de un paquete de DBMS, hasta que las pruebas demuestren la capacidad del sistema.

La tendencia de los DBMS es que la categoría de las relaciónales, será la predominante, debido a sus ventajas sobre las otras dos.

La ventaja principal que se tiene con el enfoque relacional sobre los de red y jerárquico esta en la simplicidad de su representación en la estructura lógica de la base de datos y la flexibilidad para establecer relaciones de datos por medio de campos de conexión. Todas las entidades en una base de datos relacional están representadas como tablas separadas y no están colocadas en ninguna jerarquía fija, como es el caso de los arboles o estructuras plex. La mayoría de los usuarios finales probablemente entienden las tablas mucho mejor de lo que entenderían una estructura compleja de red.

El enfoque relacional hace posible el alcanzar mayor independencia de los datos usando campos de conexión en lugar de apuntadores para enlazar registros relacionados en diferentes archivos ( o relaciones ).

Una característica única del sistema relacional es su independencia de trayectorias de entrada/salida. Ya que una base de datos relacional consta de una colección de tablas separadas, cualquier tabla o relación, se puede accesar directamente sin necesidad de accesar otras relaciones basadas en una estructura de datos fija, tales como un árbol o una red El sistema relacional permite recuperar una tabla ( o un conjunto de registros en vez de un solo registro ) con una sola instrucción del Lenguaje manipulador de datos DML.

### 5.1.2 Arquitectura de un sistema relacional de base de datos.

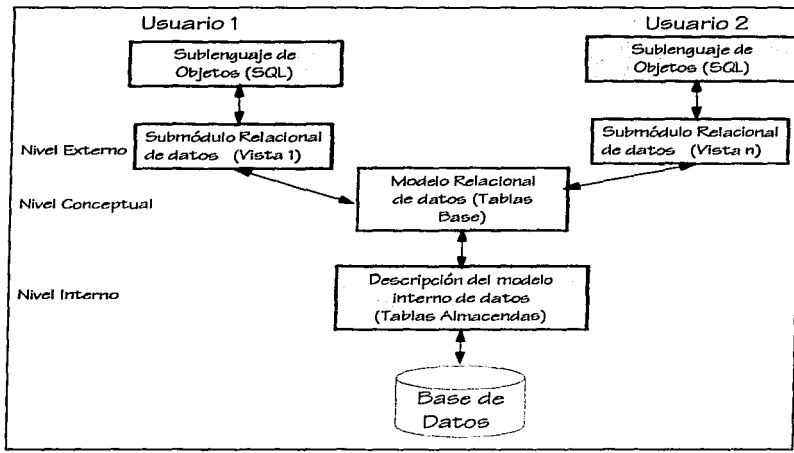



**Figura 5.2 Arquitectura de un sistema relacional de base de datos.**

Si observamos la figura 5.2 hay tres niveles de abstracción y los componentes que integran la arquitectura relacional.

Esquema de almacenamiento: en el nivel interno, cada tabla base de implanta como un archivo lógico almacenado (que puede ser conformado por varios archivos físicos). Para las recuperaciones sobre las llaves principal o secundaria se puede establecer uno o más índices para accesar el archivo.

Modelo relacional de datos. En el nivel conceptual, el modelo relacional de datos esta representado por una colección de relaciones almacenadas (también llamadas tablas base). Cada registro de tipo conceptual en un modelo relacional de datos se implanta como un archivo diferente.

Submodelo de datos. Los esquemas externos de un sistema relacional se llaman submodelos relaciónales de datos; cada uno consta de uno o más escenarios o vistas para describir los datos solicitados por una aplicación cualquiera. Una vista puede incluir datos de una o más tablas.

Cada programa de aplicación está provisto de un buffer, que es un área de trabajo del usuario donde el DBMS puede depositar los datos recuperados de la base para su procesamiento, o puede guardar temporalmente sus salidas antes de que el DBMS las escriba en la base de datos. Es decir se puede ejecutar un programa y si este falla la información de la base de datos no es tocada.

Sublenguaje de datos. Es un lenguaje de manejo de datos para el sistema relacional. Dos sublenguajes fueron propuestos originalmente; a) álgebra relacional, b) cálculo relacional. Estos fueron propuestos por Codd que además demostró que ambos lenguajes son relacionalmente completos y equivalentes, así que cualquier relación que puede derivarse de una o más tablas de datos también se puede derivar con un solo comando del Sublenguaje. Por tanto el modelo de operación de entrada/salida en un sistema relacional se puede procesar en la forma de una tabla a al vez en lugar del procesamiento tradicional de un registro a la vez. Dicho de otra forma, se puede recuperar una tabla en vez de un solo registro con la ejecución de un comando del Sublenguaje de datos.
El álgebra relacional es un lenguaje de procedimientos de alto nivel que permite el uso de operadores para derivar la tabla deseada en diversos pasos desde las tablas base originales en el modelo relacional, o también otras tablas derivadas intermedias. Por otro lado, el cálculo relacional es un potente lenguaje de consulta que permite al usuario la recuperación de datos mediante el establecimiento de condiciones de consulta sin necesidad de codificar en forma detallada los pasos de la recuperación, y aunque no es tan accesible el álgebra relacional para el usuario, como el cálculo relacional, proporciona los conceptos básicos para el desarrollo de lenguajes relaciónales de manejo de datos, Los dos principales sublenguajes relaciónales son : SQL (Structured Query Language) y QBE (Query by Example), están implementados con el mismo espíritu que el cálculo relacional.

El SQL se puede usar no solo como DML sino también como DDL, para definir tablas base, escenarios o vistas y los archivos almacenados en los niveles conceptual, externo e interno, es además flexible y sus comandos, pueden emitir interactivamente o desde un programa principal. SQL es el lenguaje que más DMBS relaciónales han adoptado en la actualidad.

Usuarios. Éstos realizan accesos a la base de datos mediante instrucciones del sublenguaje de datos o en el caso de los usuarios programadores, algunas veces puede hacer uso del SQL, las limitaciones son a causa de la seguridad interna del DBMS.

Los DBMS en su gran mayoría constan de dos subsistemas principales, que son:

El sistema de indagación de almacenamiento (Research Storage System (RSS)) y el Sistema Relacional de Datos (Relational Data System (RDS)).

El RDS proporciona la interfaz del usuario externo, que soporta las estructuras de datos tabulares y los operadores sobre esas estructuras, que generalmente son SQL, y el RSS proporciona al RDS una interfaz de registros almacenados.
El RDS es una interfaz similar al sistema de control de la base de datos. Sus funciones más características son el análisis sintáctico y la optimización. Así cuando el sistema recibe una instrucción SQL, se analiza gramaticalmente con base a la gramática del lenguaje para determinar si pertenece o no a el y si es así, cual es su estructura. El optimizador de RDS escogerá una trayectoria eficiente de acceso y el mandato SQL se traducirá a los comandos adecuados para el llamado de las rutinas de control de ejecución de RDS:

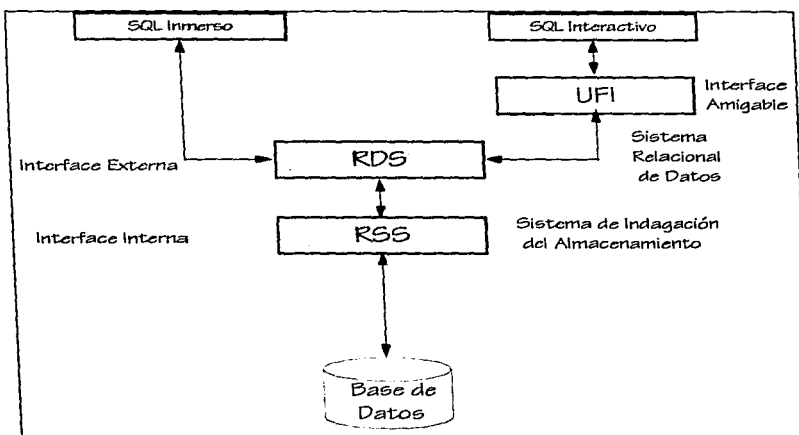



**Figura. 5.3 Subsistemas de un sistema relacional.**

## 5.2 Programación del Sistema.

La petición de servicios de procesamiento por computadora se puede codificar o expresar en un lenguaje natural, tal como el inglés. Hoy en día existe un conjunto de llamadas técnicas de cuarta generación que han cambiado el término "Lenguaje de programación " . Más que codificar, los que desarrollan algún tipo de sistema de gestión pueden hoy en día escribir los resultados deseados, en lugar del procedimiento deseado. Así se genera automáticamente el código fuente en un lenguaje de programación convencional.

Cuando se considera como un paso del proceso de ingeniería del software, la codificación es una consecuencia general del diseño. Sin embargo, las características del lenguaje de programación y el estilo de programación pueden afectar profundamente a la calidad y al mantenimiento del software.

### 5.2.1 El proceso de traducción.

El proceso de codificación traduce una representación del software dada por un diseño detallado a una realización en un lenguaje de programación. El proceso de traducción continúa cuando un compilador acepta el código fuente como entrada y produce como salida un código objeto dependiente de la máquina. Más tarde la salida del compilador es traducida a código máquina, es decir las instrucciones reales que dirigen la lógica cableada o microprogramada de la unidad central de proceso.

El proceso inicial de traducción, del diseño detallado al lenguaje de programación, es un punto fundamental dentro del contexto de la ingeniería del software. En el proceso de traducción puede aparecer "ruido" de muchas formas distintas. La interpretación equivocada de las especificaciones del diseño o las restricciones de un lenguaje de programación pueden conducir a un código fuente muy liado o "amarrado" que resulte difícil de probar y mantener. Más sutilmente, las características de un lenguaje de programación pueden influir en la forma de pensar, propagando diseños de software y estructuras de datos innecesariamente limitados.

Las características del lenguaje tiene un impacto directo sobre la calidad y la eficiencia de la traducción.

Desde un punto de vista ingenieril las características de los lenguajes de programación se centran en las necesidades que puede tener un proyecto específico de desarrollo de software. Aunque se podrían derivar esotéricos requerimientos para el código fuente, se pueden establecer un conjunto general de características de ingeniería: 1) facilidad de traducción del diseño al código; 2) eficiencia del compilador; 3) portabilidad del código fuente ; 4) disponibilidad de herramientas de desarrollo y 5) facilidad de mantenimiento.

El paso de codificación comienza tras haber definido, revisado y modificado en caso necesario el diseño. En teoría, la generación de código fuente a partir de las especificaciones del diseño detallado deberá ser algo directo.

El grado de facilidad de la traducción del diseño al código proporciona una indicación de como se aproxima un lenguaje de programación a la representación de diseño.

Por otro lado los rápidos avances en velocidad del procesador y densidad de memoria ha comenzado a disminuir la necesidad de "código super eficiente", aunque muchas aplicaciones requieren programas más rápidos y a la medida. Las críticas actuales a los compiladores de lenguajes de alto nivel van dirigidas a la incapacidad de producir código ejecutable rápido y justo a la medida.

Hay que considerar también que un código fuente tiene portabilidad cuando puede ser transportado de un procesador a otro y de un compilador a otro sin ninguna o muy pocas modificaciones.

Es importante documentar el diseño y así proporcionar un fundamento para la lectura y comprensión y con ello, el código fuente final puede ser modificado de una manera más propia.

La disponibilidad de herramientas de desarrollo puede acortar el tiempo requerido para la generación del código fuente y puede mejorar la calidad del código.

### 5.2.2 Elección de un lenguaje.

Para nuestro caso la elección de un lenguaje requiere de la consideración de todas las características antes vistas. Sin embargo, el problema asociado con la elección puede desaparecer si sólo se dispone de un lenguaje o si el cliente demanda uno en particular, o la organización donde se va a desarrollar existe un estándar sobre los lenguajes que pueden utilizarse.

Entre los criterios que se aplican durante la evaluación de los lenguajes disponibles están 1 ) área de aplicación general ; 2 ) complejidad algorítmica y computacional; 3) entorno en el que se ejecuta el software; 4) consideraciones de rendimiento; 5) complejidad de las estructuras de datos, y 6) disponibilidad de un buen compilador o compilador cruzado. El área de aplicación de un proyecto es el criterio que más se aplica durante el proceso de selección del lenguaje.

En nuestro caso los lenguajes disponibles pueden ser de alto nivel, o de cuarta generación o de programación orientada a objetos. Para fines prácticos solo hago descripciones generales para que el diseño del sistema pueda ser instalado en cualquier plataforma y en cualquier lenguaje.

FORTRAN, RMCOBOL , Fox-Pro, Power Builder, Visual Basic son sólo ejemplos en los cuales puede ser desarrollado para equipos pequeños y mini con un usuario o multiusuario en red.

### 5.2.3 Clases de lenguajes.

Existen cientos de lenguajes de programación que han sido aplicados en uno u otro momento de algún esfuerzo serio de desarrollo de software. Dado que cualquier clasificación de lenguajes de programación permanecerá abierta a debate, solo presentare el desarrollo de las cuatro generaciones de lenguajes que corresponden a la evolución histórica de los lenguajes de programación. Ver figura 5.5.

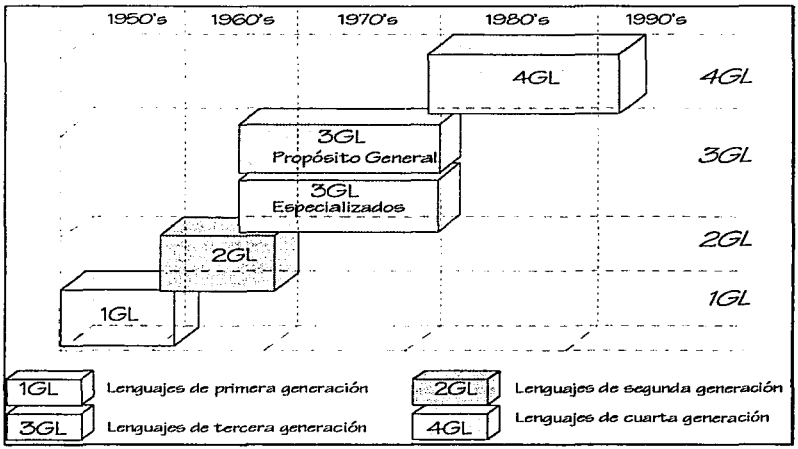



**Figura 5.5 Generaciones de lenguajes de programación.**

### 5.2.3.1 Primera generación de lenguajes.

La primera generación de lenguajes surgió cuando se codificaba a nivel de lenguaje de máquina. El código más conocido es el lenguaje ensamblador, el cual representa a la primera generación de lenguajes. Estos lenguajes son totalmente dependientes de la máquina ya que están en el menor nivel de abstracción con el que se puede representar un programa.

Según el número de arquitecturas de procesadores existen los mismos lenguajes ensambladores. Desde el punto de vista ingenieril de software, el uso de estos lenguajes será cuando un lenguaje de alto orden no satisfaga los requerimientos o no este disponible.

### 5.2.3.2 Segunda Generación de Lenguajes.

Esta generación fue desarrollada a finales de la década de los 50's y principios de los 60's y ha servido como base de todos los lenguajes de programación modernos. La segunda generación de lenguajes esta caracterizada por su amplio uso, la enorme cantidad de bibliotecas de software y la gran familiaridad y aceptación. Hay pocas discusiones sobre que FORTRAN COBOL ALGOL y BASIC son lenguajes de base debido a su madurez y su aceptación.

FORTRAN ha permanecido a lo largo de casi 30 años de críticas y sigue siendo el primer lenguaje de programación en el ambiente científico y de ingeniería. COBOL al igual que FORTRAN , han alcanzado madurez y es el lenguaje aceptado como "standard" para aplicaciones de procedimiento de datos comerciales. Aunque el lenguaje ha sido a veces criticado por su falta de unidad, tiene excelentes posibilidades de definición de datos, es muy auto documentado y proporciona soporte para un gran rango de técnicas algorítmicas relativas al procesamiento de datos en los negocios. ALGOL es el predecesor de muchos lenguajes de tercera generación y ofrece un repertorio extremadamente rico de construcciones procedimentales y de tipificación de datos. ALGOL ha sido extensamente usado en Europa, pero ha encontrado poca ayuda en América (exceptuando el entorno académico). BASIC es un lenguaje originalmente diseñado para enseñar programación en modo de tiempo compartido. el lenguaje parecia destinado a quedarse obsoleto a principios de los 70, pero con la llegada de la computadoras personales ha vuelto a renacer.

### 5.2.3.3 Tercera Generación de Lenguajes.

Los lenguajes de tercera generación, también denominados lenguaje de programación moderna o estructurada, están caracterizados por sus potentes posibilidades procedimentales y estructuración de datos. Los lenguajes de esta clase se pueden dividir en dos categorías, lenguajes de alto orden de propósito general y lenguajes especializados.

**Lenguajes de alto orden de propósito general**

Estos lenguajes tiene diferentes tipos de datos y permiten la tipificación de datos, permiten el uso de subprogramas y permiten una amplia gama de estructuras de control. El lenguaje de alto orden de propósito general más antiguo es ALGOL, el cual ha servido como modelo para otros lenguajes de esta categoria. Sus descendientes, PL/1, Pascal, Modula-2, **C** y Ada, han sido adoptados como lenguajes con potencial para un gran espectro de aplicaciones.

**Lenguajes especializados.**

Los lenguajes especializados están caracterizados por su usual formulación sintáctica que ha sido especialmente diseñada para una aplicación particular. Actualmente se usan cientos de lenguajes especializados. En general estos lenguajes tienen una base de usuarios mucho menor que los lenguajes de propósito general. Entre estos lenguajes usados por la comunidad de ingeniería del software están Lisp, PROLOG, Smalltalk, APL y FORTH.

### 5.2.3.4 Lenguajes de Cuarta Generación.

A lo largo de la historia del desarrollo de software siempre se ha intentado generar programas de computadora en cada vez mayores niveles de abstracción. Los lenguajes de la primera generación trabajan a nivel de instrucciones de máquina, el menor nivel de abstracción posible. Los lenguajes de segunda y tercera generación han subido el nivel de representación de programas de computadora, pero aún hay que especificar distintos procedimientos algorítmicos perfectamente detallados. Durante la década pasada, los lenguajes de cuarta generación (L4G) elevaron aún más el nivel de abstracción.

Los lenguajes de cuarta generación combinan características procedimentales y no procedimentales, o sea, el lenguaje permite al usuario especificar condiciones con sus correspondientes acciones (componente procedimental) mientras que pidiendo al mismo tiempo al usuario que indique el resultado deseado (componente no procedimental) para encontrar los detalles procedimentales aplicando su conocimiento del dominio específico.

Existen tres categorías de L4G que son:

**Lenguaje de petición:** La gran mayoría de los L4G se desarrollaron para ser usados con aplicaciones de bases de datos. Estos lenguajes de petición permiten al usuario manipular de manera compleja la información contenida en las bases de datos previamente creadas.

**Generadores de Programas:** Aunque más sofisticados que los anteriores, más que basarse en una base de datos predefinida, un generador de programas permite al usuario crear programas en un lenguaje de tercera generación usando menos instrucciones. Estos lenguajes de programación de muy alto nivel hacen un fuerte uso de la abstracción de datos y procedimientos. Desgraciadamente para los que trabajan en el dominio de sistemas y de ingeniería, la mayoría de estos generadores de programas son para aplicaciones de sistemas de información de negocios y generan programas en COBOL.

**Otros L4G:** De los L4G más comunes son los dos anteriores, sin embargo, existen otras categorías. Los lenguajes de soporte a la toma de decisiones permiten que los programadores lleven a cabo una gran variedad de análisis, que van desde los simples modelos de hojas de cálculo bidimensionales hasta los sofisticados sistemas de modelos estadísticos y de investigación de operaciones. Los lenguajes de prototipos se han desarrollado para asistir en la creación de prototipos facilitando la creación de interfaces para el usuario y de diálogos, además de proporcionar medios para el modelado de datos.

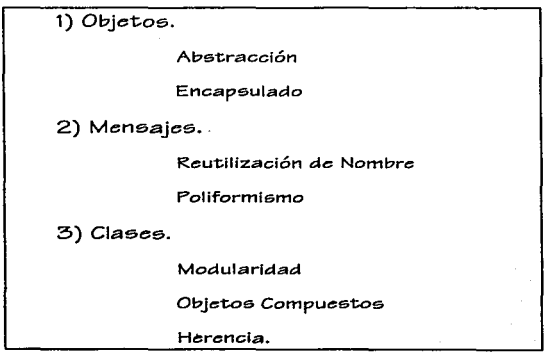
## 5.3 Programación Orientada a Objetos (POO).

Actualmente existe un nuevo enfoque para la construcción de software, que ofrece una mejor forma de construir sistemas pequeños o a gran escala que sean confiables, flexibles, de fácil mantenimiento y capaces de evolucionar para satisfacer los requerimientos del cambio, este es conocido como la tecnología de Programación Orientada a Objetos (**POO**).

La **POO** se maneja con varios conceptos que para su explicación los agruparemos relacionalmente en partes o módulos de la siguiente manera:

1) Objetos, 2) Mensajes y 3) Clases.

La siguiente figura (Fig. 5.4) representa la estructura para representar los conceptos de la **POO**.



### 5.3.1 Objetos.

Los sistemas orientados a objetos pensando abstractamente,. son una colección de dispositivos virtuales independientes que se comunican entre sí, cada uno de ellos con su propia estructura interna. A estos dispositivos virtuales se les llama objetos.

Así , un objeto es la abstracción de una entidad tangible del mundo real o de un concepto teórico. El objeto tiene estado, comportamiento e identidad. Entonces un objeto puede ser: una casa, un automóvil, un avión, un catálogo.

El objeto se caracteriza por tener estado (peso, posición, altura) y comportamiento (aceleración, cambio de lugar). El modificar el comportamiento provoca ya sea la ejecución de alguna acción, o el cambio en el estado del objeto. Un objeto tiene identidad porque es único y diferente a cualquier otro.

El estado de un objeto se modela a través de un conjunto de valores variables, que representan la abstracción del objeto y sus características en el tiempo, a los cuales llamamos datos. El comportamiento se modela a partir de un conjunto de funciones que llevan a cabo ciertas tareas que modifican el estado del objeto y se les conoce como métodos. Los datos y los métodos se manejan como un todo indivisible.

Los datos internos de un objeto son inaccesibles a cualquier otro objeto o programa. Este concepto es llamado encapsulado y es uno de los principales de POO.

La figura 5.5 muestra como los métodos del objeto protegen a sus datos de ser alterados por cualquier elemento externo. La única forma de modificar el estado del objeto es a través de un método, manteniendo así la integridad de los datos.

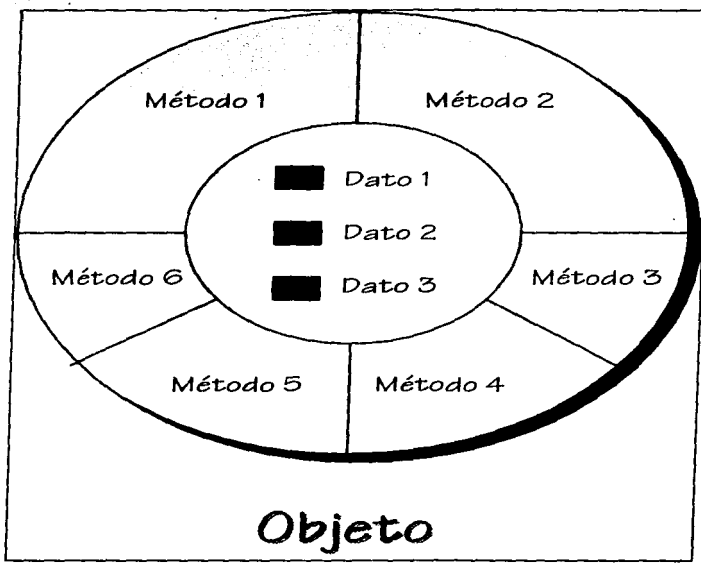



**Figura 5.5 Representación de un Objeto.**

**5.3.2. Mensajes-.**

Los objetos se comunican entre sí a través de mensajes como se ilustra en la figura 5.6.

Los mensajes modifican el comportamiento del objeto receptor, provocando que realice cierta opción o que cambie su estado. Por ejemplo, al crear un objeto, o al preguntar por el valor de un dato.

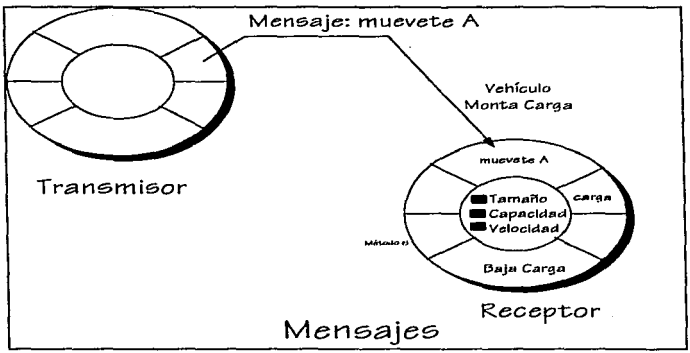



**Figura 5.6 Representación de un mensaje.**

El Poliformismo es otro concepto de la POO, el cual se refiere a que cada objeto reacciona de manera particular a un mensaje que se le envía.

Supongamos que se manda el mensaje para que todos los animales coman "comer", todos lo harán de forma particular. Si pongo otro animal, el sabrá como comer, enviándole solo el mismo mensaje: comer. De esta manera el flujo normal del programa no cambia, volviéndose mucho más sencillo y de fácil mantenimiento.

### 5.3.3 Clases.

Una clase es la definición abstracta de un grupo de objetos con datos y métodos comunes.

Los objetos se crean mediante las instancias de un clase. Mientras que un objeto es la identidad concreta que existe en el tiempo y el espacio, una clase sólo representa una abstracción, o sea la esencia del objeto.

Una clase define las propiedades y métodos que comparten todas sus instancias, pero cada objeto tiene sus propios valores únicos; como se muestra en la figura 5.7, cada montacargas tiene un tamaño, capacidad, velocidad máxima y contenido específico. Aunque cada objeto tiene sus propios datos, sí comparte los métodos de la clase con los demás objetos.

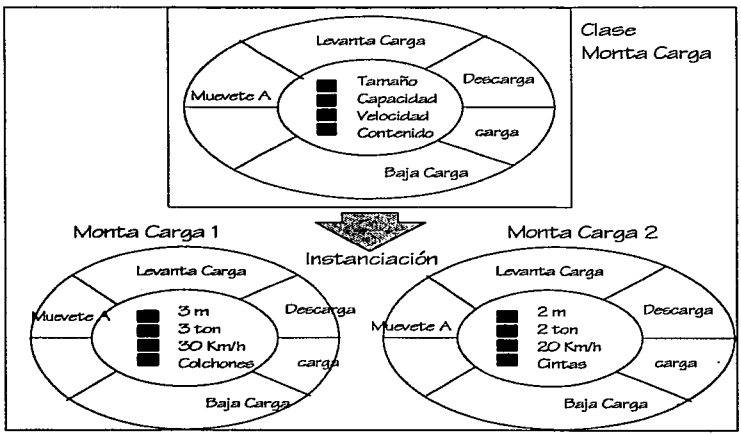



**Figura.5.7 Ejemplo de la clase monta carga.**

Un objeto siempre es la instancia de una clase, pero una clase bien puede no producir objetos. A este tipo de clases que sólo sirven para expresar características comunes a un grupo de clases, se l llama clase abstracta y cobra sentido en una jerarquía de clases.

Las clases traen orden a los objetos organizándolos en jerarquías de clases - herencia- y en objetos compuestos -objetos que contienen otros objetos - . Estos conceptos se explican más adelante.

### 5.3.4 Métodos de Organización.

Según Yourdon y Coad, los métodos de organización que existen son tres:

1. *Identidad. Identificar los objetos por medio de sus atributos (estado) y comportamiento. Por ejemplo un león.*
2. *Organización Parte De (Objetos Compuestos). Descomponer al objeto en las partes que lo forman. Por ejemplo un león tiene ojos, patas y cabeza.*
3. *Organización Tipo De (Jerarquía de Clases). Identificar la relación que tiene con otros objetos de características semejantes. Por ejemplo el león pertenece a los felinos.*

La aplicación que tiene estos métodos es la POO es la siguiente:

El punto 1. Identidad, se explicó en el tema de los objetos, donde la actividad más importante es la de identificación y definición de los objetos.

Los puntos 2 y 3 están íntimamente relacionados con las clases y los utilizamos para simplificar nuestra apreciación del mundo.

Organización Parte De. cuando le explicamos un niño pequeño las partes que tiene un automóvil, le decimos: Un automóvil tiene puertas, ruedas, volante y motor. La Organización Parte De explica al objeto en término de sus componentes. A este tipo de objetos se les denomina objetos compuestos.

Organización Tipo De. Al hacer la descripción de un sofá , a un niño que sabe lo que es una silla, no le explicamos desde la nada, sino que le decimos: "...un sofá es como una silla, sólo que más cómodo y más grande..."

Las clases actúan como una plantilla agrupando objetos relacionados entre sí con propiedades y comportamientos similares.

En la Organización Tipo De, las clases se ordenan de una forma jerárquica, en una taxonomía de superclases (padres) y subclases (hijos). Es precisamente esta taxonomía la que define la jerarquía de clases.

Con las subclases es posible definir objetos más especializados, conforme desciende la jerarquía. Las superclases expresan características comunes entre clases, entre más cerca nos encontremos de la raíz, las clases son más generales; entre más cerca de las hojas, las clases son más particulares.

### 5.3.5 Herencia.

Hablando en términos generales una subclase hereda todos los datos y métodos de la superclase a la que pertenece. La subclase puede adicionar datos y métodos a los que heredo de su padre, aunque también puede modificar los métodos heredados. La figura 5.8 muestra el concepto de jerarquía de clases y de herencia. Todos los transportes tiene un cierto tipo de combustible, una capacidad de carga y transportan bienes o personas de un lugar a otro. Todas las subclases de transporte heredan estas características.

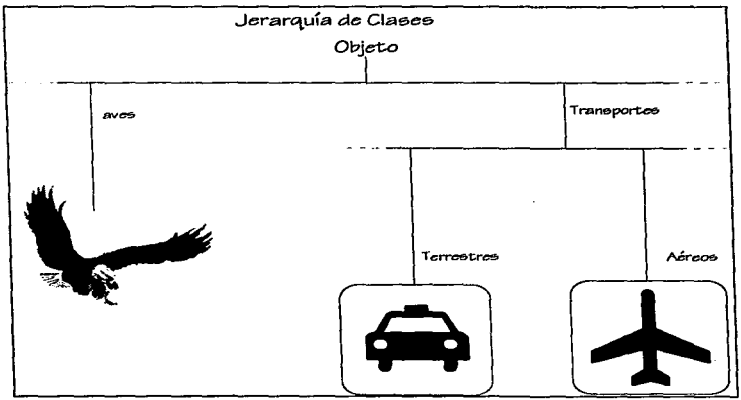



**Figura 5.8 Jerarquía de clases**

Además de la rama de transportes, todos los transportes terrestres comparten algunas características comunes, de la misma manera que lo hacen los transportes

aéreos. Los transportes terrestres se mueven en tierra firme, mientras que los aéreos por aire.

Como ejemplo de las ventajas de la herencia, es que si se desea otra rama de transporte, sea transporte marítimos, su definición es mucho más fácil, porque ya tenemos todas las características que comparten los transportes. Solo necesitamos definir la información específica sobre esta rama, sea por ejemplo, que los transportes marítimos se desplazan en el agua.

La herencia constituye la principal forma de reutilizar código en la programación orientada a objetos POO.

La programación orientada a objetos es una forma más natural de ver las cosas.

**5.4 Programación Orientada a Eventos.**

La programación orientada a eventos está basada en lo que es la programación orientada a objetos POO. Los objetos usados son los componentes que frecuentemente aparecen en la interfaz gráfica de Windows, estos objetos son por ejemplo las ventanas, los botones, los botones radiales, las cajas de listas, las líneas de edición, los textos estáticos. Mientras que los métodos asociados a los objetos, son precisamente los eventos. entre los diferentes eventos que encontramos están la creación del objeto, la destrucción del objeto, la modificación del texto, el hacer "clic" sobre el objeto, hacer doble "clic".

La programación orientada a eventos es fácilmente explicada, utilizando la filosofía que ofrece la programación orientada a objetos para hacer una analogía entre ambas.

**5.5 Como desarrollar el sistema de estimación y control de proyectos.**

Dentro de los objetivos de este trabajo no esta el de mostrar como desarrollar particularmente en un lenguaje el sistema, "... No des un pez, al que tiene hambre, enséñale a pescar...", por ello, la construcción del sistema queda abierta para cualquier ambiente de hardware y software, donde cada quién puede desarrollarlo particularmente y aún más puede aumentar o reducir el número de módulos, aquí mostrados, ya que la pauta está puesta.

Los que tenga herramientas de software de tercera. generación ( COBOL, FORTRAN 77 , FORTRAN 77X, BASIC ) le será de utilidad enfocar los conceptos de base de datos al manejo de archivos secuenciales e indexados.

En el caso de utilizar DBMS sencillos como Dbase IV Plus , Clipper, Fox Pro u otros similares, el desarrollo del sistema será aún más fácil.

Para él que desea implementarlo en un equipo multiusuario y con un DBMSR, le será práctico utilizar las herramientas CASE con que cuentan estos Administradores de Bases de Datos Relaciónales, ya que según el tipo que se utilice, los procesos en línea ON-line como son conocidos, podrán ser rápidamente desarrollados, utilizando los subsistemas con que cuentan para esto. Los reportes podrán desarrollarse con gran facilidad y rapidez, además de poder ser modificados según necesidades particulares dentro de la aplicación real, según el giro de la empresa donde sea aplicado el sistema.

Con el uso de estos manejadores el módulo de seguridad puede ser reforzado, utilizando las herramientas para la generación de seguridad con que cuentan.

Los diálogos para altas, bajas, cambios y consultas de los módulos de Partidas, Básicos y Productos, Estimaciones, Infraestructura, El contabilizador, Seguridad, Estadísticas, se podrán desarrollar con los estándares de estas aplicaciones, los cuales ofrecen además, herramientas para el control de los cambios y nuevas versiones, ya que como se menciono un sistema al liberarse debe continuar con el refinamiento, ya que no es posible que de una sola vez, quede perfecto.

El desarrollo en POO puede ser en VisualBasic o Power Builder o C++ o en cualquier otra aplicación, ya que debido a la versatilidad con que se puede integrar a los sistemas ya existentes dentro de una organización, el sistema puede ser otro más dentro de los sistemas transaccionales de una organización y estar en interfaz con otros sistemas tantos administrativos, como de diseño, por solo decir unos ejemplos.

$\Omega$

# CAPÍTULO 6

# Pruebas y Puesta en Marcha.

**Pensar es el trabajo más difícil que existe. Quizá sea ésta la razón por la que haya pocas personas que lo practiquen.**

**Henry Ford.**

# CAPITULO 6

## Pruebas y Puesta en Marcha.

### 6.1.- Pruebas del Sistema.

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

La prueba presenta una interesante anomalía para los ingenieros de software ya que en las fases anteriores de definición y de desarrollo, lo que intenta el ingeniero es construir software partiendo de un control abstracto y llegando a una implementación tangible. Posteriormente al llegar la etapa de pruebas el ingeniero crea una serie de casos de prueba, que intentan demoler y acabar con el software que ha sido construido. De hecho, la prueba es uno de los pasos de la ingeniería de software que se puede ver como destructivo en lugar de constructivo.

#### 6.1.1. Flujo de Información de la Prueba.

El flujo de información para la prueba sigue el esquema que se describe en la figura 6.1. Se dan dos clases de entradas al proceso de prueba: 1) una configuración del software que incluye la especificación de requerimientos del software, la especificación de diseño y el código fuente; y 2) una configuración de prueba que incluye un plan y procedimiento de prueba, casos de prueba y resultados esperados.

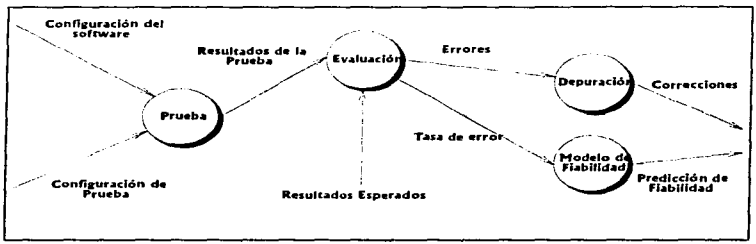



***Figura 6.1*** Flujo de Información de la Prueba.

Se lleva a cabo la prueba y se evalúan los resultados, o sea, se comparan los resultados de la prueba con los esperados. Cuando se descruben datos erróneos esto implica que empiece una depuración.

A medida que se va recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse un grado cualitativo de calidad y fiabilidad del software. Si se encuentran serios errores que se encuentren son fácilmente corregibles, se puede decir que la calidad y fiabilidad del software son aceptables.

Cualquier producto de ingeniería puede ser probado de una de dos forma: 1) conociendo la función especifica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa; 2) conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que "todas las piezas encajan"; o sea que, la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comportado de forma adecuada. La primera aproximación se denomina de la caja negra y la segunda de la caja blanca.

### 6.1.2 Técnicas de Prueba.

Cuando se considera el software de computadora, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

La prueba de caja blanca del software se fundamenta en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejecuten casos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o afirmativo.

### 6.1.2.1 Prueba de la Caja Blanca.

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. mediante los métodos de prueba de caja blanca el ingeniero puede derivar casos de prueba que: 1) garanticen que se ejercitan por lo menos una vez todos los caminos independientes de cada módulo; 2) se ejercitan todas decisiones lógicas en sus caras verdaderas y falsa; 3) se ejecutan todos los bucles en sus límites y con sus límites operacionales, y 4) se ejercitan todas las estructuras de datos internas para asegurar su validez.

Quizá en este punto sea valido preguntarse ¿por qué gastar tanto tiempo en minusciosidades en lugar de gastarlo en el aseguramiento de haber alcanzado los requerimientos del programa?. Pero las justificaciones se dan a continuación.

Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa. Los errores tienden a reproducirse en nuestro trabajo cuando diseñamos e implementamos funciones, condiciones o controles que se encuentran fuera de lo normal.

A menudo creemos que un camino lógico tiene pocas posibilidades de ejecutarse cuando, de hecho, se puede ejecutar de forma regular.

Los errores tipográficos son aleatorios. Cuando se traduce un programa a código fuente de un lenguaje de programación, es muy probable que se den algunos errores de escritura. Muchos serán descubiertos por los mecanismos de comprobación de sintaxis, pero otros permanecerán indetectados hasta que comience la prueba.

Las pruebas de caja blanca se centran en la estructura de control del programa. Se derivan casos de prueba que aseguren que durante la prueba se han ejecutado por lo menos una vez todas las sentencias del programa y que se ejercitan todas las condiciones lógicas.

La prueba del camino básico, es una técnica de las denominadas de caja blanca, hace uso de grafos de programa (o matrices de grafo) para derivar el conjunto de caminos linealmente independientes que aseguren la cobertura. La prueba de bucles complementa a otras técnicas de la caja blanca proporcionando un procedimiento para ejercitar bucles de distintos grados de complejidad.

### 6.1.2.2 Prueba de la Caja Negra.

Los métodos de prueba de la caja negra se centran en los requerimientos funcionales del software. La prueba de la caja negra permite al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales de un programa. La prueba de la caja negra no es una alternativa a las técnicas de la caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de la caja blanca.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías: 1) funciones incorrectas o ausentes; 2) errores de interfaz, 3) errores en estructura de datos, o en accesos a base de datos externas; 4) errores de rendimiento; 5) errores de inicialización y determinación.

A diferencia de las pruebas de caja blanca, la prueba de caja negra tiende a ser aplicada durante fases posteriores de prueba. Ya que la prueba de la caja negra intencionadamente ignora la estructura de control, concentra su atención en el dominio de la información.

Las pruebas de caja negra son diseñadas para validar los requerimientos funcionales sin fijarse en el funcionamiento interno de un programa. Las técnicas de prueba de la caja negra se centran en el dominio de la información de un programa de forma que se proporcione una cobertura completa de prueba . La partición equivalente divide el dominio de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. El análisis de valores límite, prueba la habilidad del programa para manejar datos que se encuentren en los límites aceptables. Los diagramas de causa efecto se incluyen en una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. Finalmente, la prueba de validación de datos asegura que los datos interactivos (conducidos por órdenes) sean procesados correctamente.

### 6.1.3 Estrategia de Prueba del Software.

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie planificada que lleva a una construcción correcta del producto de software. Y lo que es más importante, una estrategia de prueba de software proporciona un plano para el programador, para la organización de control de calidad y para el cliente, un plano describe: los pasos a llevar a cabo como parte de la prueba; cuando se deben de planificar y realizar estos pasos, y cuanto esfuerzo, tiempo y recursos serán requeridos.

Una estrategia de prueba de software debe ser suficientemente flexible para promover la creatividad y la adaptabilidad necesarias para adecuar la prueba a todos los grandes sistemas basados en software. Al mismo tiempo, la estrategia debe ser suficientemente rígida para promover un razonable seguimiento de la planificación y la gestión a medida que progresa el proyecto.

Existen y se han propuesto varias estrategias de prueba de software en distintos trabajos, en común todas estas pruebas proporcionan al ingeniero de software una plantilla para la prueba y todas tienen las siguientes características generales:

La prueba comienza en el nivel del módulo y trabajan "hacia afuera" hacia la integración del sistema completo.

En diferentes puntos son adecuadas a la vez distintas técnicas de pruebas.

La prueba la lleva a cabo el que desarrolla el software y un grupo de prueba independiente.

La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

La estrategia de prueba que se sigue para el software producido se puede explicar al observar la figura 6.2 ( Pasos de prueba de software). La prueba, en el contexto de la ingeniería de software, realmente es una secuencia de cuatro pasos.

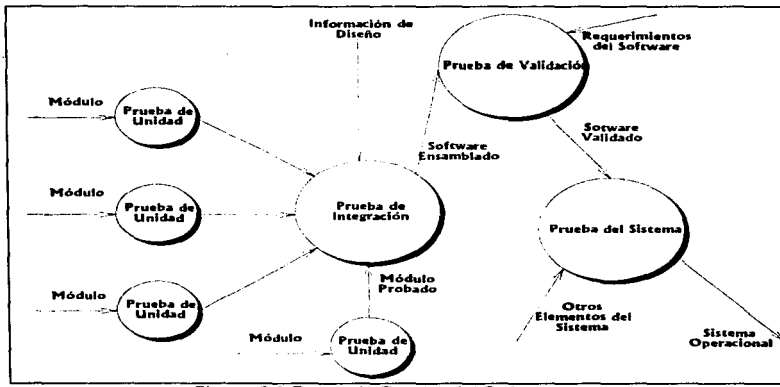



*Figura 6.2 Pasos de Prueba del Software.*

Inicialmente la prueba se centra en cada módulo individual, asegurando que funcionen adecuadamente como una unidad. De ahí el nombre de prueba de unidad. La prueba de unidad hace uso intensivo de las técnicas de prueba de la caja negra, ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores. A continuación se deban ensamblar o integrar los módulos para formar el paquete de software completo. La prueba de integración se dirige a todos los aspectos asociados con el doble problema de verificación y de construcción del programa. Durante la integración, las técnicas que más prevalecen son las de diseño de casos de prueba de la caja negra, aunque en algunos casos se pueden llevar a cabo una pocas pruebas de caja blanca para asegurarse de que cubren los principales caminos de control. Por último se deben comprobar los criterios de validación. La prueba de validación proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Durante la validación se usan exclusivamente técnicas de prueba de caja negra, el fin de la prueba es la prueba del sistema y aquí se verifica que cada elemento ajusta de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total.

### 6.1.4 Prueba de Unidad.

La prueba de unidad centra el proceso de verificación en lo que es la menor unidad del software "El Módulo". Usando la descripción del diseño, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad siempre está orientada a la caja blanca, y éste paso se puede llevar a cabo en paralelo para múltiples módulos.

En la prueba de unidad se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad del programa que esta siendo probada. Se examinan también las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente y conservan su integridad durante todos los pasos de ejecución del algoritmo. Se prueban además las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y finalmente se prueban todos los caminos de manejo de errores.

### 6.1.5 Prueba de integración.

El reto que aquí se analiza es cuando los módulos independientes interaccionan en conjunto, ya que los datos se pueden perder en una interfaz, un módulo puede tener un efecto adverso sobre otro, las subfunciones cuando se combinan pueden o no producir la función principal deseada, la imprecisión aceptada individualmente puede crecer hasta niveles inaceptables; y las estructuras de datos globales pueden presentar problemas.

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que al mismo tiempo se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es escoger los módulos probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Para llevar a cabo la prueba se utiliza el proceso de integración incremental en donde el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir, es más probable que se puedan probar completamente las interfaces y se pueda aplicar una aproximación de prueba sistemática.

La prueba para el software producido se llevó a acabo mediante integración descendente, la cual como su nombre implica, empieza la construcción y la prueba con los módulos indivisibles ( módulos de niveles más bajos de la estructura del programa). Dado que los módulos son integrados de abajo hacia arriba, el procesamiento requerido de los módulos subordinados siempre está disponible y se elimina la necesidad de resguardos.

### 6.1.6 Pruebas de validación.

Una vez que el software se encuentra ensamblado y se han corregido los errores de interfaces, la prueba que sigue es la de la validación. La validación se logra cuando el software funciona de acuerdo con las expectativas razonables del cliente, las expectativas razonables están definidas en la especificación de requerimientos.

La validación de software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requerimientos. El plan de prueba traza las pruebas que se deben llevar a cabo y un procedimiento de prueba para definir los casos de prueba especificos que serán usados para demostrar la conformidad con los requerimientos. Tanto el plan como el procedimiento están diseñados par asegurar que se satisfacen todos los requerimientos funcionales y que se alcanzan todos los requerimientos de rendimiento.

### 6.1.7 Prueba del Sistema.

Cuando es software es incorporado a otros elementos del sistema como son un nuevo hardware o información, se realiza una serie de pruebas de integración del sistema y de validación.

La prueba del sistema realmente está conformada por una serie de pruebas diferentes cuyo propósito primordial es la ejercitar profundamente el sistema. Aunque cada prueba tiene un propósito distinto, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

Las pruebas mencionadas anteriormente son: 1) la prueba de recuperación, para forzar a un fallo de software de muchas formas y verificar que la recuperación se lleva a cabo apropiadamente; 2) prueba de seguridad, verifica que los mecanismos de protección incorporados al sistema lo protegerán, de hecho, de la penetración impropia; 3) prueba de resistencia, para enfrentar a los programas con situaciones anormales; 4) pruebas de rendimiento, la cual esta diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

**6.2.- Puesta en Marcha del Sistema.**

La puesta en marcha del sistema es el proceso de colocar el sistema de información desarrollado en la operación normal dentro de las actividades de la empresa. Esto se realiza una vez que el producto de software se encuentra totalmente desarrollado y aprobado por las especificaciones con las cuales fue creado, entonces lo único que falta es insertarlo en el ambiente operativo de la empresa e iniciar las operaciones.

La puesta en marcha del sistema es de suma importancia para el éxito del mismo, ya que un inicio de operaciones deficiente o mal implementado puede causar el fracaso inicial del producto desarrollado. Para el proceso de puesta en marcha se deben considerar todos los factores que influyen en la operación de un sistema como los siguientes: factor humano (capacitación para los usuarios), factor procedimental ( implementación de procedimientos para administrar la operación del sistema), factor hardware (comunicaciones, cableado, etc.), factor de conversión de sistemas (migración de información entre el sistema anterior y el nuevo).

Los factores que consideramos para la puesta en marcha del sistema son los factores de capacitación a los usuarios y la carga inicial de datos, dichos factores los consideramos parte del proyecto de desarrollo e implantación del producto de software Sistema de Control de Proyecto.

### 6.2.1. Capacitación a Usuarios.

Dado que la capacitación es un factor para el éxito del producto de software se pone un especial énfasis en dicho factor, ya que incluso el mejor sistema puede tener éxito o fallar debido a la forma en que se opera y se utiliza. Por lo anterior la calidad con la que se de la capacitación a los usuarios que manejaran al sistema ayuda o dificulta el éxito de la puesta en marcha de un sistema de información. Las personas que se verán involucradas con el sistema deben conocer en máximo detalle las funciones que realiza el sistema, así como la forma de utilizarlo y lo que hace y deja de hacer.

La mayor parte de la capacitación de los usuarios radica en la operación del sistema mismo aunque en ocasiones la capacitación abarca el empleo del equipo y en estos casos los usuarios deberán recibir primero la capacitación respecto a la operación del equipo.

Dentro de la capacitación, se destacan las actividades de manejo de datos que reciben la máxima atención durante el proceso de capacitación de usuarios y los cuales son los siguientes, la entrada de nuevos datos (es la inserción de nuevos datos a los ya existentes) y la consulta de datos (búsqueda de datos almacenados previamente). La parte más importante del uso del sistema incluye el conjunto de actividades descritas anteriormente, por lo que la capacitación ocupa la mayor parte de su tiempo en estas actividades.

Los dos aspectos más importante que se manejarán en la capacitación de usuarios son: 1) la familiaridad con el sistema de procesamiento, es decir, el equipo que se usa; 2) la capacitación en el empleo de la aplicación, es decir, en el software que recibe los datos, los procesa y genera el resultado. Es importante resaltar que si se tienen carencias en cualquier aspecto de la capacitación probablemente se terminará por hacer fracasar la utilización del sistema en la operación de la empresa.

### 6.2.1.1. Capacitación en el uso del Sistema de Estimación y Control de Proyectos.

Dado que en la empresa donde será empleado el sistema de estimación y control de proyectos, se visualizan cuatro tipos de usuarios, entonces la capacitación será enfocada a estos cuatro tipos: capturistas, técnicos, administrativos y directivos.

La capacitación a los usuarios incluye aspectos de manejo de equipo de cómputo, manejo de impresoras y aspectos del funcionamiento del producto de software. Es importante insertar procedimientos administrativos internos con la parte del sistema que soporta la operación. Los temas generales que forman parte de la capacitación básica para todos los usuarios del sistema de estimación y control de proyectos son:

Aspectos básicos del manejo de PC's e impresoras.
Ciclo operativo de la estimación y manejo de proyectos.
Aspectos generales del sistema: antecedentes, esquema básico operativo, impacto en la operación, apoyo al control de actividades de la empresa.

Así los anteriores puntos integran la capacitación básica, con ellos adquieren los conocimientos mínimos para considerarse parte activa del proyecto de estimación y control de proyectos.

Una vez impartido el curso básico a los integrantes de la empresa que estarán involucrados con el sistema se agruparán de acuerdo a la actividad que tendrán entorno al sistema, es decir, de acuerdo al tipo de usuario que hayan sigo catalogados. Para cada grupo de usuarios se desarrollan cursos de capacitación, de manera general, los puntos tratados en la capacitación de cuatro tipos diferentes de usuarios son:

Usuario Tipo Capturista.

Manejo de PC's e impresora.
Actividades propias dentro del ciclo de operación diaria, procedimientos administrativos a seguir y relación con el sistema de información.
Módulo de Seguridad.
Módulo de Básicos y Productos.
Módulo de Partidas.
Módulo de Inventarios.
Módulo de Subcontratistas.
Módulo de Estimaciones.
Módulo de Infraestructura.
Módulo de Procesos Batch.

Usuario Tipo Técnico.

Manejo de PC's e impresora.
Actividades propias dentro del ciclo de operación diaria, procedimientos administrativos a seguir y relación con el sistema de información.
Módulo de Seguridad.
Módulo de Básicos y Productos.
Módulo de Partidas.
Módulo de Inventarios.
Módulo de Subcontratistas.
Módulo de Estimaciones.
Módulo de Infraestructura.
Módulo de Guías Contabilizadoras.
Módulo de Procesos Batch.

Usuario Tipo Administrativo.

Manejo de PC's e impresora.
Actividades propias dentro del ciclo de operación diaria, procedimientos administrativos a seguir y relación con el sistema de información.
Módulo de Seguridad.
Módulo de Básicos y Productos.
Módulo de Partidas.
Módulo de Inventarios.
Módulo de Subcontratistas.
Módulo de Estimaciones.
Módulo de Infraestructura.
Módulo de Estadísticas.
Módulo de Guías Contabilizadoras.

Usuario Tipo Directivo.

Módulo de Seguridad.
Módulo de Básicos y Productos.
Módulo de Partidas.
Módulo de Inventarios.
Módulo de Subcontratistas.
Módulo de Estimaciones.
Módulo de Infraestructura.
Módulo de Estadísticas.
Módulo de Guías Contabilizadoras.

La capacitación se imparte durante la instalación del software a la empresa y está incluye aspectos tanto teóricos como prácticos usando como ejemplos tipo de información con la que se trabaja a diario, pero sin llegar éste a ser real.

Posteriormente para el inicio de operaciones con el sistema, se vuelve a impartir la capacitación a los usuarios, pero con la diferencia de que los ejemplos ya contienen la información verdadera y real con la que se estará trabajando.

# CONCLUSIONES.

La falta de control sobre los proyectos, provoca desajustes en la operación de las empresas, la terminación de los proyectos en lo común toman más tiempo del estimado originalmente. El problema reside en que, rara vez quienes tiene la responsabilidad de controlar el proceso o los procesos tienen de modo confiable el grado de avance y menos aún de cuales pueden ser sus fechas de terminación, ya no se diga de los costos estimados y en secuela de los reales. La solución plantea varias alternativas, dentro de las cuales tenemos:

1. Utilizar metodologías tradicionales de control.

2. Utilizar una metodología propia, con mezcla de:
   - Investigación de operaciones
   - Estadística aplicada
   - Sistemas de CAD-CAM.
   - Técnicas de evaluación económica.

El presente trabajo propone la última alternativa, inculcar el conocimiento en cada uno de los lectores, para que de acuerdo a sus necesidades, logre una herramienta propia.

Apuntando concretamente a los capítulos, las técnicas descritas, para el análisis, diseño, desarrollo, pruebas e implantación del sistema, son las más comúnmente utilizadas por los profesionales de la informática, sin que por ello, contengan un alto grado de dificultad para no ser empleadas por un usuario con perfil profesionista.

Los conceptos sobre Administradores de Base de Datos, su arquitectura, el como diseñar una base de datos, los tipos de ellas, sus componentes, darán apoyo a que sumados estos a los conocimientos que se tenga en el funcionamiento del negocio, proporcionen resultados mejores, más cercanos a lo óptimo.

Las técnicas de capacitación ayudaran a que el conocimiento del sistema que se desarrolle, sea transmitido lo más propio posible.

En general el éxito del sistema se debe a muchos factores, pero principalmente si el análisis y diseño son de excelente calidad, ya que éstos sientas las bases para que el desarrollo también lo sea, y además teniendo las facilidades que proporciona una interfaz gráfica; esto hace que el producto realizado sea de gran calidad y beneficio para la empresa que lo utiliza. Dejando los cimientos para que, en el futuro se puedan ampliar los horizontes y límites del sistema.

Al hacer uso de las principales ventajas que nos ofrece la metodología de generación de prototipos y combinarla con la metodología de cuarta generación, nos permitió elaborar un producto mejor, que si seguimos forzosamente una metodología individual.

El planteamiento de combinar las metodologías mostradas para el desarrollo del sistema fue de gran importancia, ya que la combinación de los modelos tradicionales del software con la utilización de herramientas de cuarta generación, nos permitirá generar el software que se necesite, para la empresa que se quiera.

En el desarrollo del sistema existen además de los aspectos ya descritos, otros factores que intervienen y que no son considerados muchas veces, como es el **ingenio** y astucia para atacar y resolver problemas, en otras palabras, enfrentar los retos que presenta la programación del sistema, existen ocasiones en que el problema tiene más de una solución, pero el escoger la mejor muchas veces depende de la implementación que haya realizado el programador, ya que la mejor solución, no es vislumbrada desde el principio, pero será la mejor, la que en el momento se enfoque a las necesidades del sistema y esto es fundamental para el desempeño eficiente del mismo.

El uso de un sistema informático, del tipo transaccional para la estimación y control de proyectos, se sustenta en los avances de la ingeniería de software y de hardware, sumándose a las técnicas tradicionales, dándonos los siguientes beneficios:

- Seguridad en la información.
- Rapidez en el flujo de información.
- Exactitud en los datos.
- Contar con una metodología con datos cuantificables.
- Ventaja tecnológica, respecto a la competencia.
- Reducción en los costos de administración del proyecto.

Por último la justificación total del trabajo cumple con la máxima para la que debe ser creado un sistema que es, la de aumentar la productividad y no generar actividades inútiles. **"...Hay que producir más, con menos..."**.

# BIBLIOGRAFÍA


**Hillier/Lieberman:**
**Introducción a la investigación de Operaciones**
**Mc. Graw Hill 1990. México, D.F.**

**James A. Senn**
**Análisis y Diseño de Sistemas de Información.**
**Mc. Graw Hill México, D.F. 1994.**

**Edward Yourdon**
**Análisis Estructurado Moderno I/E**
**Prentice Hall Hispanoamerica, S.A. 1993 México.**

**C. J. Date**
**Introducción a los sistemas de bases de datos**
**Addison Wesley Iberoamericana, S.A. 1986 Wilmington, Delaware, E.U.A.**

**Charette, Robert N.**
**Software Engineering Enviroments Concepts and Technology**
**Prentice Hall 1988 USA.**

**Pressman, Roger.**
**Ingenieria de Software, Un enfoque práctico,**
**Prentice Hall España 1977.**

**Kendall, Kenneth E. Kendall, Julie E.**
**Análisis y Diseño de sistemas.**
**Prentice Hall, México 1991.**

**Koch George**
**ORACLE Manual de Referencia**
**Osborne McGrawHill México 1993.**

**Fairley, Richard.**
**Ingeniería de Software**
**McGraw Hill, 1era. edic. México, 1988.**

**Cullinnet**
**IDMS Manual de Referencia para Diseño de Bases de Datos**
**USA 1989.**

**Date, C. J.**
**Introduction to Database Systems**
**4th. ed. Addison Wesley, USA. 1986.**

**Schjetdnan Dantán, Mario**
**Ruta Crítica al alcance de todos.**
**UNAM Ciudad Universitaria**

**Canavos. George C.**
**Probabilidad y Estadistica Aplicaciones y Métodos.**
**Mc. Graw Hill México, D.F.**

**Alexander Hamilton**
**Como controlar su tiempo**
**Moder Business Reports New York USA**

**Ryan-Mc Farlan Co.**
**RMCOBOL 85**
**Ryan-Mc Farlan Co. USA**

**Martin L. Shooman**
**Software Engineering**
**Mc Graw Hill Singapore**

**Martín Marmolejo G.**
**Inversiones IMEF**
**México, D.F.**

**Enid Squire**
**Diseño de sistemas**
**Representaciones y Servicios de Ingeniería. México D.F.**

**Richard Barker**
**CASE Method Task and Deliverables**
**Addison Wesley USA**

**James Martin Carma McClure**
**Structured Techniques**
**Prentice Hall USA**

Ω