UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A C A T L A N

"LA IMPORTANCIA DE UTILIZAR UNA METODOLOGIA EN EL DESARROLLO DE SISTEMAS . . . UN CASO PRACTICO"

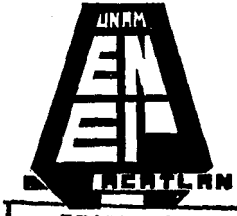
TRABAJO DE MEMORIAS DEL DESEMPEÑO PROFESIONAL

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A

JUAN CARLOS RENDON AGUILAR



STA. CRUZ ACATLAN, EDO. DE MEX., FEBRERO 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

ÍNDICE

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

INTRODUCCIÓN

Pocos son los ámbitos en la vida diaria del hombre que han quedado fuera del desarrollo de la computación y de la tecnología de las comunicaciones. Es gracias a ésto que el manejo de datos e información han cobrado enorme importancia dentro de nuestra sociedad.

En un principio, las computadoras eran dispositivos de gran tamaño, pero con el paso del tiempo se han transformado en pequeños aparatos que fácilmente pueden ser usados sobre un escritorio. Los costos de Hardware fueron reduciéndose en forma considerable debido principalmente al desarrollo de los superconductores. Algunas aplicaciones avanzadas de la computación como lo son la Inteligencia Artificial, Multimedia, etc., son ya económicamente viables. El Hardware que se requiera para satisfacer grandes necesidades de cómputo puede ser conseguido a precios relativamente moderados. Como resultado de este impresionante desarrollo, se tiene que las economías de las empresas e incluso de las naciones, depende cada vez más de las computadoras.

Para mala fortuna, no podemos hablar de la misma forma del Software, puesto que contrario a lo ocurrido con el hardware, aquel se inició a un costo muy bajo (incluso se regalaba en la compra de equipo). Actualmente, los costos del software se han incrementado en una forma impresionante, incluso llegan a valer hasta un 80% o más del costo total del sistema

La historia del desarrollo de sistemas se remonta hasta el nacimiento de las primeras computadoras, cuyos programas eran escritos en lenguaje máquina. Posteriormente, al surgir el concepto de programa almacenado y con la creación de algunos lenguages denominados de "alto nivel", el desarrollo de programas comenzó una nueva etapa de desarrollo, en la cual se creaban programas pequeños que funcionaban en computadoras con una capacidad muy limitada. Con el paso del tiempo, y gracias al desarrolo de la tecnología de la computación, el tamaño de los programas se fue incrementando en forma impresionante, pero también aumentaron los problemas. Debido a eso, fue necesario implementar una serie de estrategias que facilitaran el desarrollo de dichos programas.

Desafortunadamente la comercializacion de los equipos de cómputo y de su software, ha venido propiciando que día con día se desarrollen programas de ínfima calidad a un precio muy elevado.

El objetivo principal de este trabajo, es el de mostrar la importancia de utilizar una metodología en el desarrollo de sistemas, básandome en una serie de experiencias personales adquiridas a lo largo mi vida profesional y durante mi estancia en Japón, tomando un curso teórico-práctico sobre ingeniería de sistemas, y retomando algunas otras opiniones de personas que han dedicado muchos años de su vida al desarrollo de sistemas.

En el capítulo uno, se presentan una serie de conceptos básicos que se utilizarán a lo largo del trabajo.

En el capítulo dos, se hace un estudio comparativo de los diferentes modelos de ciclo de vida y de las metodologías más utilizadas para el desarrollo de sistemas.

En el capítulo, tres se propone una metodologia alterna y se explican sus principios básicos, así como la descripción de un caso práctico, que servirá de base para ejemplificar el proceso de diseño descrito en el capítulo cuatro.

En el capítulo, cuatro se explican las etapas del diseño de procesos y de archivos a fín de mostrar lo importante que es hacer una documentación clara del sistema bajo estudio.

En el capítulo cinco, se hace un análisis de las perspectivas que presenta el desarrollo de sistemas

CAPÍTULO I

"Haz las cosas tan SENCILLAS como se pueda, pero no SIMPLES".

ALBERT EINSTEIN

CAPÍTULO I

CONCEPTOS BÁSICOS EN EL DESARROLLO DE SISTEMAS

Los sistemas prevalecen en la naturaleza, formando una característica esencial de todas las cosas. Ningún acercamiento a las organizaciones puede ser más natural que el enfoque de sistemas[1].

Las organizaciones humanas pueden ser vistas como una red de sistemas interdependientes diseñados para desarrollar actividades vitales para el hombre[2]. Es imposible encontrar sistemas perfectos puesto que vivimos en un mundo dinámico, siempre cambiante, en donde la perfección es un término muy vago y un objetivo muy variable. Por tanto, todas las organizaciones (empresas públicas, privadas, etc.), requieren de la estructura de un sistema y de procedimientos que les permitan realizar sus funciones en forma adecuada, que les guíe en las operaciones que ejecutan día con día[3]. En pocas palabras, vivimos en un mundo lleno de sistemas, sin importar que sean biológicos o hechos por el hombre.

---

1 Fitzgerald, J. "Fundamentals of Systems Analysis"
2 Hitachi, Co. "Current System Study and Analysis"
3 Hitachi, Co. "An Overview of O.A."

I.1. SISTEMAS

Como se mencionó anteriormente los sistemas son de suma importancia en nuestra sociedad, pero, ¿qué es un sistema?.
Existen muchas definiciones que parten de los mismos principios, algunas de éstas pueden ser:

"Un sistema es la unión de varios componentes interrelacionados y organizados para completar un conjunto de metas"[4] Fig. I.1,

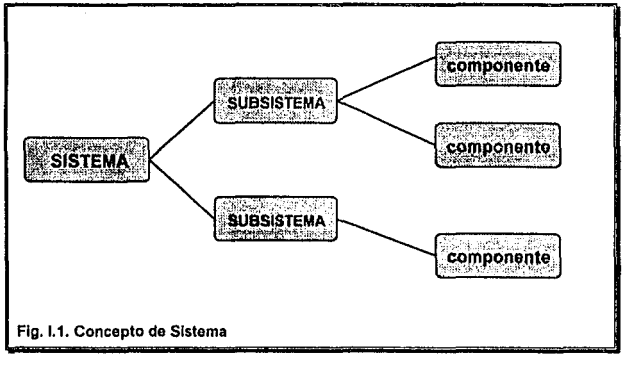


Fig. I.1. Concepto de Sistema

"Un sistema es un conjunto de elementos y procedimientos intimamente relacionados que tienen como propósito el logro de determinados objetivos"[5] Fig. I.2.

[4] Hitachi, Co. "Introduction to Systems Design"
[5] Hora, J. L. "Introducción a la Informática"

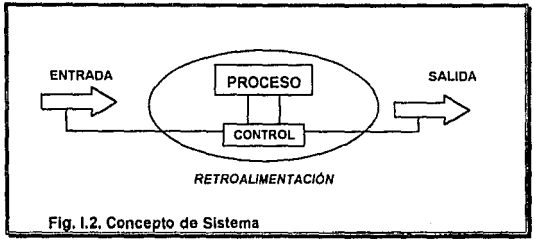


Fig. I.2. Concepto de Sistema

También, "un sistema puede ser definido como una red de procedimientos interrelacionados que son unidos para realizar una actividad o para completar un objetivo específico"[6].

Un procedimiento es una serie de instrucciones precisas que explican:

- ***Qué se va a hacer***
- ***Quién lo hará***
- ***Cuándo será hecho***
- ***Cómo se hará***

Básicamente, todos estos conceptos son muy parecidos entre sí, todos hablan de conjuntos de elementos y procedimientos relacionados que nos permiten tomar decisiones sobre cómo lograr los objetivos de la organización.

Existen muchos tipos de sistemas:

- ***Sistemas de transporte***
- ***Sistema nervioso***
- ***Sistema solar***
- ***Ecosistemas***
- ***Sistemas de información***

Este trabajo se enfocará a los sistemas que requieren del procesamiento de la información para realizar una toma de decisiones adecuada. Sistemas que requieran de registro, procesamiento y reportes de información significativa, es decir, que

[6] Fitzgerald, J. "Fundamentals of Systems Analysis"

constituyan un medio para obtener información veraz y oportuna que nos permita tomar cursos de acción, ésto es, tomar decisiones.

Dichos sistemas se presentan principalmente en empresas o instituciones, en la organización de las mismas, las cuales pueden ser vistas como un sistema, siendo sus partes integrantes, subsistemas.

I.2. INFORMACIÓN

En la actualidad la información es cada vez más necesaria en todas nuestras actividades cotidianas, vivimos en una sociedad orientada hacia la información, todas nuestras actividades están completamente vinculadas a este concepto, principalmente en aquellas que intervienen directamente en nuestro desempeño profesional, en la oficina, en el taller etc., la información es parte fundamental de una organización.

En los últimos años, el enorme desarrollo que han tenido las organizaciones, demanda grandes cantidades de información. Además, las empresas en la actualidad requieren de tomar decisiones cada vez más precisas y con mayor rapidez. Es por eso que se ha dedicado mucho tiempo a investigar y desarrollar nuevas y mejores técnicas de procesamiento.

Para llevar a cabo una buena toma de decisiones, es necesario que la información sea concisa, clara y que se cuente con ella en el momento en que se requiera. De esta forma se le ayuda a la persona encargada de la toma de decisiones a formarse con suficiente anticipación una idea clara y completa de la situación actual que vive la organización y pueda tomar objetivamente las decisiones adecuadas.

Podríamos definir a la información como:

"Un conjunto de datos debidamente procesados y convertidos a una forma útil; es el conocimiento derivado del análisis de los datos"[7] ***Fig. I.3.***

[7] Senn, J. "Sistemas de Información para la Administración"

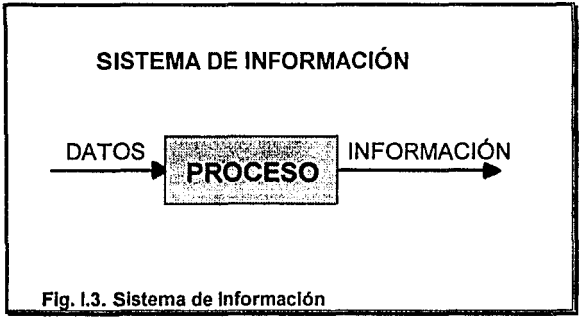


Fig. I.3. Sistema de Información

Como hemos visto, la información es una parte esencial de cualquier compañía que tiene como objetivo desarrollar una efectiva y eficiente actividad en los negocios. Las organizaciones que cuentan con mayor información, generalmente presentan un alto índice de desarrollo. Podemos decir que una empresa bien informada, es aquella que tiene un funcionamiento más efectivo. Como vemos, la información es tan importante que se le ha considerado como un sistema, al que se le denomina **sistema de información**, cuyo fin es procesar información y soportar las actividades de los sistemas de negocios.

I.3. SISTEMAS DE INFORMACIÓN Y DE NEGOCIOS

Una empresa es un sistema de divisiones de trabajo asociadas con jerarquías en varias secciones, todas unidas para cargar con las actividades diarias y dirigidas a completar las metas corporativas[8] Fig. I.4.

[8] Hitachi, Co. "Current System Study and Analysis"

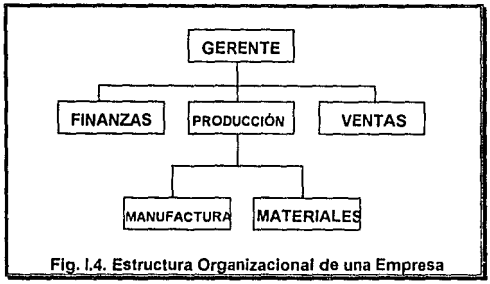


Fig. I.4. Estructura Organizacional de una Empresa

Las actividades de los negocios tales como dirección y administración se completan a través del uso de la información. Estas actividades también forman un sistema, conocido como sistema de negocios, los cuales cuentan con recursos humanos y físicos y se ven como una colección de funciones para completar las metas de la corporación Fig. I.5.

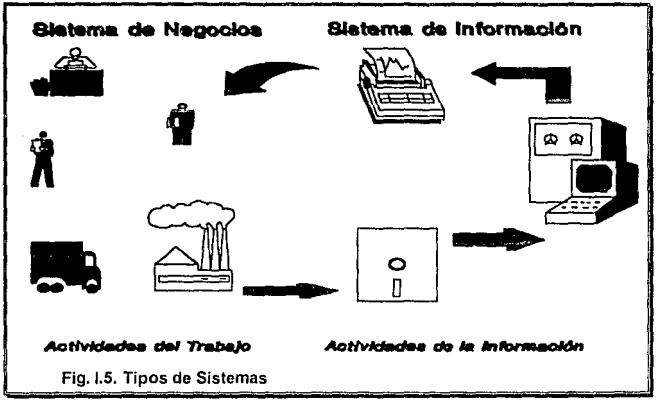


Fig. I.5. Tipos de Sistemas

Los sistemas de información proveen de medios para completar metas dentro de las divisiones de trabajo de una empresa; divisiones tales como mercadotecnia, producción y contabilidad. El diseño de un eficiente sistema de información no se puede lograr sin antes analizar las actividades actuales del negocio.

I.4. INFORMÁTICA

La informática se encarga de estudiar el diseño y la utilización de equipos, sistemas y procedimientos que permiten captar y procesar los datos adecuadamente para obtener la información útil en la toma de decisiones. Se puede considerar a la informática como una ciencia completamente relacionada con la toma de decisiones.

Dentro de la informática, existen varias ramas:

- ***EL PROCESAMIENTO ELECTRÓNICO DE DATOS***
- ***LA ADMINISTRACIÓN DE CENTROS DE CÓMPUTO***
- ***EL ANÁLISIS Y DISEÑO DE SISTEMAS***

Esta última es de suma importancia ya que en ella se estudia y se diseñan los sistemas y procedimientos con relación a la toma de decisiones.

Estudiar y analizar las actividades actuales es el primer paso en el desarrollo de sistemas, a travéz del cual se puede establecer un modelo del nuevo sistema capaz de implementar mejoras sugeridas en las actividades. El nuevo sistema propuesto es entonces gradualmente elaborado mediante el uso de computadoras y otras herramientas de procesamiento de la información.

I.4.1. OBJETIVOS DEL ANÁLISIS Y DISEÑO DE SISTEMAS

El análisis es un estudio preparatorio para llevar a cabo alguna acción. El propósito del análisis es decidir qué acciones se deben tomar y describirlas completamente.

Así, el análisis envuelve la selección de objetivos para el desarrollo del sistema, el establecimiento de criterios para el éxito y la predicción de progresos a través de metas, cerrando el ciclo entre requerimientos e implementación.

Dentro de los objetivos del análisis está el construir un modelo del sistema, esto es, crear una descripción de lo que se desea y de lo que eventualmente se construirá. Además, se debe asistir en el entendimiento del sistema tomando en cuenta relaciones y estructuras complejas.

Una vez que se tiene un modelo del sistema y se cuenta con suficiente información, es necesario hacer un diseño lógico de las nuevas funciones, procedimientos de captura y accesos efectivos al sistema mediante el diseño de formas y pantallas, para después hacer el diseño físico que finalmente será implantado[9].

I.5. MODELANDO LOS SISTEMAS

Dado que el producto principal del análisis es el modelo del sistema actual, podemos tener diferentes tipos de modelos; ya sea que se hable de sistemas de información, de negocios, etc., el modelo siempre nos describirá los aspectos físicos y funcionales del sistema actual[10].

I.5.1. MODELOS DE SISTEMAS DE INFORMACIÓN

Este tipo de modelos, generalmente representan las relaciones entre los datos y las funciones y/o procesos (es decir representan el flujo de datos), conformando así el diccionario de datos[11].

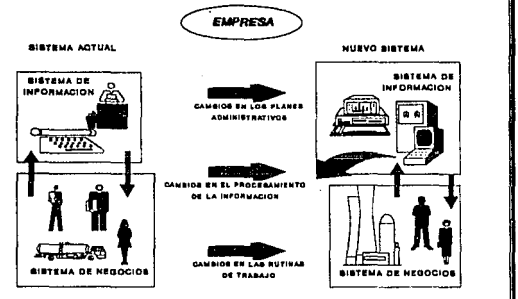
I.5.2. MODELOS DE SISTEMAS DE NEGOCIOS

Aquí principalmente se define el flujo de procesos del negocio, la estructura organizacional, el diagrama jerárquico o de funciones y los organigramas[12].

---

9 Fujitsu, Co. "Systema design Notes"
10 Yourdon, E. "Managing the Structured Techniques"
11 Hitachi, Co. "Current System Analysis and Design"
12 Hitachi, Co. "An Overview of O.A."

Fig. I.6 Tipos de sistemas



No importa el tipo del sistema del que se esté hablando, siempre es necesario definir dos tipos de modelos, el modelo físico y el modelo lógico.

I.5.3. MODELO FÍSICO

El modelo físico envuelve un exámen del sistema actual a través del análisis de sus propiedades físicas, por ejemplo, el número y el tipo de formas, sus contenidos, usos y como fluyen en el sistema.

Estos modelos, especifican el método de conducir roles o funciones, debe basarse en cómo trabaja la función en lugar de qué es, se refleja el control organizacional y debe centrarse en el trabajo y las operaciones en lugar de funciones.

Representan la entidad estudiada en cuanto a su apariencia y nos dice como operará el sistema bajo un medio ambiente técnico

I.5.4. MODELO LÓGICO

Aquí se hace una abstracción o logicalización del modelo físico en otro modelo que elimina los mecanismos o los medios a través de los cuales se realizan las actividades junto con algunos detalles físicos; por ejemplo, en un modelo físico, podemos referirnos a una forma como la "# 123, copia rosa", pero en un modelo lógico esto se transforma en "Recibo del cliente". Nótese que sólo se ha identificado la función o el propósito de la forma y se han eliminado sus características físicas.

En los modelos lógicos se eliminan las características actuales y se expresan como funciones, marcando el trabajo y las operaciones como funciones. Debe ser construido en base a lo que es la función en lugar de cómo trabaja.

Los modelos lógicos nos describen las funciones que soportará el sistema, los datos que utilizará el sistema y la relación entre las funciones y los datos.

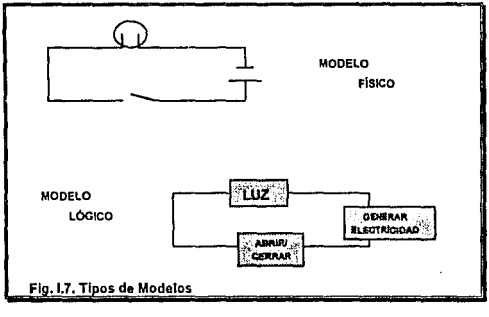


Fig. I.7. Tipos de Modelos

En el desarrollo de sistemas, es necesario definir cuatro tipos de modelos, el modelo físico del sistema actual, su equivalente modelo lógico, el cual aunado con los requerimientos del usuario, servirá de base para construir el modelo lógico del nuevo sistema y de éste, derivar el equivalente modelo físico[13].

[13] Fujitsu, Co. "System Design Reference"

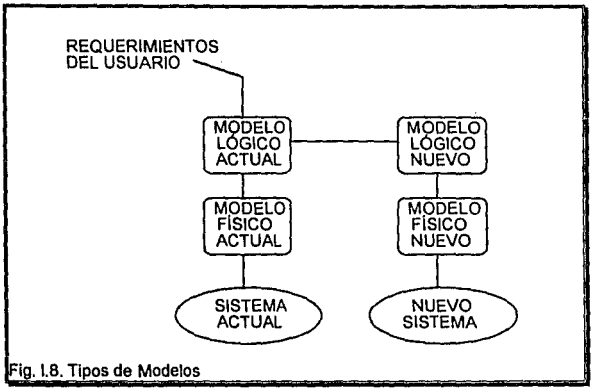


Fig. I.8. Tipos de Modelos

Las diferentes etapas dentro del modelado de sistemas, pueden ser asociadas con las diferentes fases del desarrollo de sistemas, como lo son el análisis, el diseño y la implementación, a su vez cada fase puede ser distribuida de diferente forma a lo largo del tiempo que dura el proyecto.

I.6. CICLO DE VIDA DEL DESARROLLO DE SISTEMAS

Cuando hablamos de las actividades para el desarrollo de sistemas tales como análisis y diseño, implicamos que existen algunos modelos de todas las posibles actividades dentro del desarrollo de sistemas por las cuales algunas de estas actividades han sido acomodadas en el análisis y otras en el diseño. Otros aspectos del desarrollo y mantenimiento de sistemas pueden ser tratados de manera similar. La dificultad es que

necesitamos definir qué significan esos términos. Una forma de describir estas actividades y su interrelación es utilizando el concepto de **"ciclo de vida"**, esto es, un modelo de las actividades que ocurren durante el "tiempo que dura" el sistema, empezando con el inicio del proyecto y terminando con el reemplazo del sistema. En el caso del software, generalmente se trata con el mantenimiento como fase final.

El ciclo de vida del sistema describe los estados a través de los cuales el sistema pasa desde su nacimiento hasta su muerte. El utilizar modelos equivocados, asegura dificultades, quizás fallas. Utilizando un modelo adecuado, aseguramos por lo menos la posibilidad de éxito. El principal problema es distinguir entre los modelos buenos y los malos. En la actualidad, muchos modelos de ciclos de vida han sido propuestos, implementados, redefinidos, y no hay una clara indicación de cual es el mejor.

I.6.1. DEFINICIÓN DEL CICLO DE VIDA DEL SISTEMA

El término "ciclo de vida del sistema" puede tener tantas definiciones como ingenieros de sistemas existan. De cualquier forma, muchas de estas definiciones tienen mucho en común. Algunas pueden ser:

"El ciclo de vida del sistema es un modelo utilizado para explicar y ayudarnos a entender el proceso de desarrollo y mantenimiento del sistema"[14].

"El ciclo de vida del sistema es una descomposición paso a paso del proceso de desarrollo del sistema"[15]

"El ciclo de vida del sistema es una lista de cosas por hacer".

"El ciclo de vida del sistema es una herramienta administrativa y técnica para organizar, planear, programar, y controlar las actividades asociadas al desarrollo y mantenimiento de sistemas"[16].

Estas definiciones básicas, en conjunto, dan una descripción aceptable del término de ciclo de vida del sistema. Todas ellas tienen ciertas actividades en común, y describen al ciclo de vida del sistema como:

Un modelo, plan o guía orientado a actividades y procesos para uso de quienes administran el trabajo y de quienes lo hacen.

---

[14] Peter, L. "Advanced Structured Analysis and Dseign"
[15] Sommerville, I. "Ingeniería de Software"
[16] Yourdon, E. "Managing the System Life Circle"

CAPÍTULO II

"Lo que un programador puede hacer en un año, dos programadores también lo pueden hacer en un año"

SABIDURÍA DEL SOFTWARE CONVENCIONAL

CAPÍTULO II

ANÁLISIS DE LOS DIFERENTES MODELOS Y METODOLOGÍAS EN EL DESARROLLO DE SISTEMAS

El proceso de desarrollo de sistemas consiste en una serie de pasos bien definidos que deben seguirse para poder producir sistemas bien diseñados y fáciles de mantener.

Como hemos visto, el ciclo de vida en el desarrollo de sistemas nos indica las actividades que se siguen para analizar, diseñar e implementar un sistema. Además, sabemos que existen muchas formas de acomodar dichas actividades.

Existen frecuentemente personas que desarrollan sistemas que no siguen todas estas actividades, debido principalmente a restricciones de tiempo y de presupuesto. Muchas veces se sacrifican metas de calidad y de mantenibilidad para acortar el proyecto y poder sacar rápidamente el sistema. Las más importantes como lo son el análisis de requerimientos y el diseño de especificaciones se hacen a un lado para poder iniciar inmediatamente con la implementación del código.

En la actualidad, existen muchos modelos de ciclos de vida que varían en la forma de acomodar sus actividades. Estos enfoques han sido desarrollados, refinados, implementados y utilizados por muchos profesionales en el aérea de sistemas a lo largo de la historia del desarrollo de los mismos.

No podemos emitir un juicio sobre cuál modelo es mejor y cuál es peor, pero sí podemos marcar los aspectos buenos y malos que tienen dichos modelos, y así poder formarnos un criterio y hacer una elección que vaya de acuerdo con nuestras necesidades y recursos.

II.1. EL MODELO DE CICLO DE VIDA "SLAM DUNK"

(Slam Dunk se puede interpretar como "al ahí se va")

Esta es la forma más común del ciclo de vida del sistema que se usa actualmente, no sólo en nuestro país sino en todo el mundo. La estructura básica de este modelo se reduce simplemente a la programación, se comienza programando casi tan pronto como el proyecto inicia. Muchas personas que utilizan este tipo de modelos, creen que es más fácil escribir los programas, corregirlos, depurarlos, y utilizarlos, para así presentar una aproximación de lo que quiere el usuario que haga el sistema, en lugar de perder tiempo aplicando toda una metodología.

Existen algunos factores que hacen que este enfoque siga teniendo éxito entre las personas que lo aplican. Estudios hechos durante los últimos años indican que las personas que desarrollan sistemas poseen ciertas características psicológicas comunes y únicas. Una de estas características, llamada fuerza de necesidad de crecimiento (Growth Need Strength), es una fuerte necesidad de tener hecho algo [17]. Cuando elaboramos un programa, podemos ir viendo los frutos de nuestras labores en términos de código, compilando y ejecutando los programas libres de errores. Cada uno de estos pasos es un estado de logros.

Por otro lado, si nosotros, utilizando la misma cantidad de tiempo definimos la estructura general de un sistema, detallando los requerimientos y hacemos un análisis de las necesidades, o planeamos nuestras actividades, no podemos ver algo ya hecho, no tenemos un ejemplo de logro, de ahí que tiendan a evitar dichas actividades.

Otra característica de las personas que desarrollan software es una baja fuerza de necesidad social (Social Need Strength).

Una baja fuerza de necesidad social significa que la persona es más capaz para trabajar independientemente, en lugar de formar parte de un equipo [18]. Dado que el proceso de desarrollo de sistemas funciona mejor cuando diferentes especialistas son responsables de diferentes fases pero interactuando todo el tiempo, existe un serio problema al momento de formar el equipo de trabajo.

---

[17] Yourdon, E. "Managing the System Life Circle"
[18] Peters, L. "Advanced Structured Analysis and Design"

Un factor que hace que el desarrollo de sistemas sea un poco abstracto estriba principalmente en el hecho de que el software no es físico. Nadie ha visto un programa de computadora. Hemos visto cintas de computadoras, diskettes, listados, pero los programas en sí nunca han sido vistos.

En contraste, cuando examinamos una actividad física tal como la construcción de una casa, o de un avión, podemos identificar muchos estados por los que se tendrá que atravesar. Además, podemos imaginarnos qué pasaría si se realizan cambios en cualquier estado, probablemente los estados subsecuentes se verán afectados y el programa de actividades tendrá que ser alargado. La construcción de software, sin embargo, está fuera de ese contexto, es por eso que parece relativamente simple conocer los requerimientos del cliente sin análisis ni diseño. El desarrollo de nuevos lenguajes cada vez más sofisticados, está agravando más esta situación en lugar de aliviarla, puesto que estos lenguajes, permiten implementar fácilmente un sistema o parte de él y aún más fácil el modificarlo. De hecho, algunos de estos lenguajes utilizan estructuras de control , dejando a un lado la modularización y el paso de control a través de módulos de diferentes jerarquías.

Como vemos, la metodología utilizada en este ciclo de vida consiste en "tomar una idea" y "codificarla", basándose en la teoría de que habrá muchos errores (bugs) que se tendrán que eliminar del programa y por tanto, no vale la pena desperdiciar tiempo en seguir una línea científica tal como el análisis y el diseño, fig. II.1.

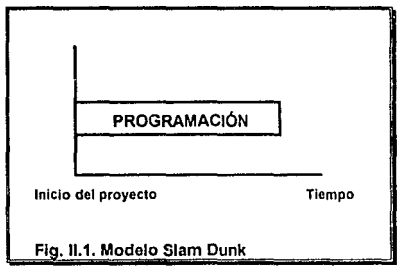


Fig. II.1. Modelo Slam Dunk

En general, podemos decir que no han sido buenas las experiencias adquiridas con este modelo, en la mayoría de los casos se desarrolla software al vapor, con una serie de deficiencias, sin tener nada que ver con las verdaderas necesidades del usuario o sub-utilizando los recursos. Este hecho ha herido seriamente la credibilidad hacia las personas que desarrollan sistemas respecto a los costos, el plan de actividades, la calidad del producto etc., puesto que como no se tiene un control de lo que se va a

hacer, es imposible determinar la duración del proyecto y por tanto, los costos se incrementan, originando que en el mejor de los casos, al tratar de evitar que dichos costos se eleven, se creen programas de baja calidad, con muchas enmendaduras.

Otro aspecto importante que repercute por la falta de control en las actividades, es que no se desarrolla ninguna documentación del sistema, no se sabe cómo esta constituido el sistema,"¿qué programas hacen qué cosa?", y peor, si el sistema está desarrollado por un "super" programador, el día que esa persona deje la empresa, y surja alguna modificación, la nueva persona asignada al proyecto, no tendrá ni la menor idea de cómo funciona ese programa y mucho menos podrá modificarlo, optando mejor por la solución más factible..."volver a programarlo".

Cabe mencionar que uno de los principios básicos de la ingeniería es el de preparar un bosquejo de las actividades que se deberán seguir, y esas actividades se deben distribuir entre los miembros del equipo, y no puede empezar la construcción hasta que ese bosquejo ha sido generado, aprobado y aceptado. Con base en ésto, nos damos cuenta de cómo este enfoque viola dichos principios.

II.2. EL MODELO BARROCO (O CÍCLICO)

Este modelo considera al desarrollo de sistemas no como un proceso lineal sino como un ciclo. Los procesos generalmente tienen un fin, pero en el desarrollo de sistemas y más en particular en el desarrollo de software, no existe. Esto ocurre debido a la etapa de mantenimiento, o a que una vez terminado el sistema, se requiera aumentar su capacidad.

Una forma de aliviar el problema de una falta de control sobre el proyecto, la falta de una disciplina y una estructura exhibidas por el enfoque "slam dunk", es el formular una verdadera disciplina. La idea es terminar con cada etapa en el desarrollo del sistema antes de iniciar con la siguiente. Este ciclo de vida marca seis etapas principales:

- ***El análisis de requerimientos***
- ***Diseño de especificaciones***
- ***Código o la implementación***
- ***Plan de prueba e integración***
- ***Entrega final***
- ***Mantenimiento***

A simple vista, esto parece ser la respuesta a los problemas encontrados en el modelo anterior ... hasta que intentemos utilizarlo. Son varios los problemas que presenta este enfoque, el principal es que se ha visto que la etapa de análisis, puede ser un proceso muy largo, es difícil el poder determinar la longitud del mismo. Esto es, podemos ir definiendo y redefiniendo los requerimientos del proyecto por tiempo indeterminado[19]. Esto se debe principalmente a que conforme más nos involucramos en el funcionamiento del sistema, descubrimos nuevas cosas, y estos nuevos descubrimientos acerca de cómo se está conduciendo actualmente el sistema, al tomarse en cuenta, casi siempre deja otros descubrimientos, los cuales dejan otros, y así sucesivamente.

Parte del problema estriba en el hecho de que la interacción que debe tomar lugar entre las actividades del análisis y diseño no está permitida. Además, lo que constituye un análisis completo no está definido y es casi imposible describirlo. Por lo tanto, el personal del proyecto trabaja hacia una meta indefinida[20].

Lo que hace que existan cambios en los requerimientos puede ser el hecho de que así como tratamos de entender estos requerimientos, aprendemos más acerca de ellos y descubrimos otras cosas que deben ser consideradas. También se debe a que, conforme el análisis se lleva a cabo, el cliente comienza a ensanchar el alcance de las consultas y comienza a ver las ventajas de la automatización. Así que los clientes comienzan a solicitar que el sistema haga más de lo que originalmente estaba planeado[21].

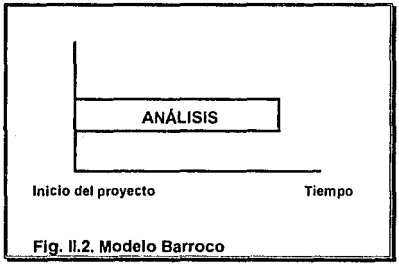


Fig. II.2. Modelo Barroco

19 Peters, L. "Advanced structured Analysis and Design"
20 Fairley, L. "Ingeniería de Software"
21 Hitachi, Co. "Introduction to System Design"

II.3. EL MODELO DE CASCADA

Este modelo, el cual es uno de los más recientes en el área de ciclo de vida del sistema, trata de corregir los desperfectos del enfoque barroco reorganizando y aventajando en tener una interrelación entre las fases. Esta interacción se logra haciendo un pequeño traslape de las etapas o fases con respecto al tiempo, lo cual le da un aspecto como de cascada. Los resultados de una fase son alimentados en la siguiente, comenzando con la fase de análisis, siguiendo con el diseño, el cual interactúa con el anterior, de tal forma que se puede definir claramente en el diseño si hace falta algo del análisis o viceversa. Es como una fábrica, en donde entra la materia prima y sale un producto terminado, moviéndose hacia otro proceso; A diferencia del modelo anterior, que presenta una especie de procesos aislados.

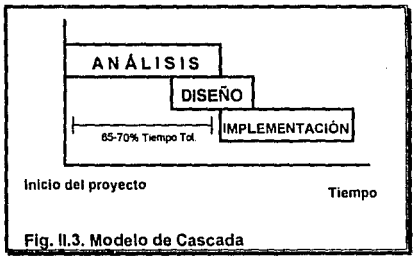


Fig. II.3. Modelo de Cascada

Este enfoque puso en práctica muchos aspectos en otros campos de la ingeniería, principalmente en la forma de utilizar las interacciones entre etapas y la forma de prevenir la tardanza del proyecto y problemas con los planes de actividades.

Uno de los aspectos que representan un problema en el uso de este enfoque, es debido a que, si tenemos que desarrollar un proyecto de tamaño relativamente largo (digamos , 2 o más años desde el inicio hasta el fin), no es sino hasta más de un año después (generalmente), cuando el cliente puede tener disponible cualquier producto del trabajo o un modelo del producto para poder examinarlo o criticarlo. De aquí que con cualquier retraso de tiempo, el proyecto puede caer en un serio problema. Es por eso que este enfoque debe ser manejado con mucho cuidado. Sabemos que la expectación del cliente siempre debe ser creciente y cambiante. Es por eso que si pasa un año o más sin tener una percepción del sistema, entonces la expectación del cliente puede dejar de crecer.

II.4. EL MODELO DE PROTOTIPOS.

Este enfoque muestra ciertos patrones semejantes al modelo de cascada. Es un complemento a la falta de objetividad en dicho enfoque. La historia del desarrollo de prototipos se remonta a hace algunos años, y es una práctica estándar en casi todas las ramas de la ingeniería. Esta práctica fue desarrollada a mediados de los 70's y su introducción sirvió de base para aplicarla en otras áreas, y se han escrito y hablado muchas cosas alrededor de ella[22] . La ventaja principal es que le da una percepción temprana a los usuarios de cómo se verá el sistema antes de que sea construido.

Este enfoque presenta ciertas desventajas, una de éstas es que el ciclo de vida y el modelo de administración podrían presentarse como el modelo slam dunk, esto pasa debido a que se acelera el desarrollo del sistema. De hecho el prototipo en sí es refinado y adoptado como el sistema entregado. El usuario puede encontrar el prototipo tan atractivo y útil que quiera adoptarlo y así obligar al equipo de desarrollo a abandonar el resto del desarrollo.

Otro problema que se presenta es el concepto de "CÓDIGO DESECHADO", el cual se utiliza cuando se desarrollan prototipos. Este concepto se refiera a que al desarrollar un prototipo, una vez finalizado el análisis y el diseño, parte de ese prototipo se desecharía (si no es que todo)[23]. Desde el punto de vista económico y metodológico, es necesario tener un considerable interés por este concepto. Los usuarios no están dispuestos a pagar por algo que se va a "desechar".

Es posible controlar este problema, todo depende de cómo se presenta el prototipo al equipo de desarrollo y qué controles de calidad se implementan, pero los riesgos están latentes. Además, se pueden utilizar los prototipos como medios para implementar un subconjunto preseleccionado de módulos con una arquitectura bien definida, desarrollada a través de un buen diseño. Esto asegura que al menos algo del código generado no será desperdiciado. Lo más que podría ocurrir es que mucho del código tendría que ser cambiado.

Este tipo de enfoques son recomendados cuando dentro del desarrollo del sistema, existe una gran interrelación con el usuario, y cuando los sistemas son de tipo interactivo y de tiempo real, puesto que el prototipo puede simplemente ser el ambiente operativo o las pantallas del sistema automatizado.

---

22 Sommerville, I. "Ingeniería de Software"
23 Cárdenas, H. "La Ingeniería de Sistemas"

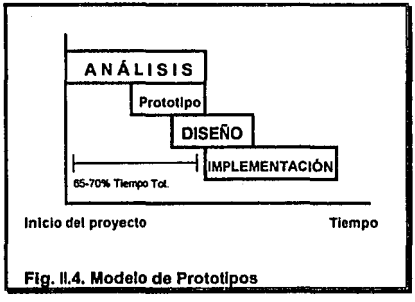


Fig. II.4. Modelo de Prototipos

II.5 EL MODELO DE FASES

Debido a que en la actualidad estamos experimentando un progreso acelerado en cuanto a los cambios en la tecnología de Hardware y Software, las corporaciones requieren de sistemas de información cada vez más grandes y más eficientes. El desarrollo de dichos sistemas grandes y técnicamente más avanzados, se ha limitado a acarrear más riesgos e incertidumbres. Una forma de impedir estos riesgos y anticipar las incertidumbres, es implementar todo el desarrollo del sistema en una serie de fases sucesivas, coordinadas por una filosofía consistente del sistema[24].

Es posible definir un pequeño ciclo de vida en cada fase, y se pueden realizar por separado. Las fases son un conjunto de pasos (o actividades) y cada paso es un conjunto de operaciones.

Este tipo de enfoques son prácticos, siempre y cuando se pretenda desarrollar un sistema de gran escala[25].

---

[24] Sommerville I. "Ingeniería de Sistemas"
[25] Fisher, A. "CASE, Using Software Developement Tools"

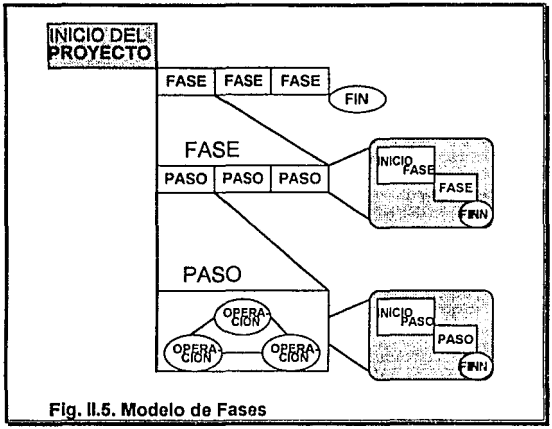


Fig. II.5. Modelo de Fases

El tomar una decisión sobre qué ciclo de vida podemos seleccionar para realizar nuestro proyecto, depende de muchos aspectos,lo más importante es que una vez definido nuestro patrón de trabajo, debemos dividir bien las actividades entre todos los miembros del equipo. Por ejemplo, en el enfoque clásico (SLAM DUNK), es muy usual que se presenten casos en donde el líder del proyecto reúne a todo su equipo, para informar que se les acaba de encomendar la tarea de desarrollar un sistema para el próximo mes, le pide a alguien que investigue qué es lo que quieren los usuarios que haga el sistema y mientras tanto los demás empiezan a codificar, por que si no, no terminaran a tiempo.

En el mejor de los casos, principalmente en algunas empresas grandes, las cuales cuentan con un departamento de sistemas, los gerentes o directores de ese departamento, implementan o adoptan algún ciclo de vida propio, o que aprendieron en algún lugar, para desarrollar sistemas o un proceso para implementar programas en sus organizaciones. Lo que en realidad han desarrollado es una especie de "recetario" que le indicará al equipo de trabajo (analistas, programadores) qué pasos tienen que seguir a través del ciclo de vida, qué tipo de documentos se tienen que generar, que archivos hay que crear, obteniendo finalmente los diagramas de flujo, estudios de costo-beneficio, y la firma de aprobación del usuario.

En una forma muy burda, se puede decir que el desarrollo de sistemas comienza cuando un usuario repentinamente piensa que requiere de un sistema computarizado. La idea surge por muchas razones, puede ser que escuchó a alguien que su sistema le estaba facilitando mucho sus actividades o ha escuchado que es lo que está de moda, y en el mejor de los casos, él mismo ve la necesidad de mejorar los procesos dentro de su organización. Una vez que esta persona ha percibido ciertas necesidades, o bien, siente el deseo de tener un nuevo sistema, lleva sus ideas al analista de sistemas, el cual reparte una serie de requerimientos funcionales a un diseñador de sistemas, y éste eventualmente reparte especificaciones de los programas al programador. En empresas grandes el programador distribuye una serie de diagramas de flujo a los codificadores quienes finalmente realizan los programas en algún lenguaje que hacen que el sistema haga lo que el usuario quiere que haga[26].

Las actividades que cada uno de los miembros del equipo tienen que realizar, se pueden explicar más específicamente: el analista se encarga de hablar con el usuario y mediante una serie de entrevistas y cuestionarios, trata de descubrir las necesidades y requerimientos del sistema bajo estudio, expresándolos de una forma tal que alguien pueda desarrollar un sistema adecuado. Esto consiste en estudiar el sistema actual, si es que existe, entrevistando a todas las personas involucradas con el funcionamiento del sistema para saber cuáles son las actividades que actualmente desempeñan, y utilizando técnicas tales como tablas de decisión para evitar que la declaración del usuario este incompleta, sea redundante o contradictoria.

El producto final del análisis de sistemas es un conjunto de especificaciones del sistema o especificaciones funcionales que describen en términos precisos, las entradas, las salidas, los algoritmos involucrados en los cómputos deseados por el usuario. Además, las especificaciones funcionales incluirán ciertas restricciones como:

- ***Tiempo mínimo de respuesta del sistema***
- ***Costo de mantenimiento***

Idealmente, los requerimientos funcionales no deben especificar decisiones orientadas al diseño computacional tales como: número de tareas, pasos, regiones, particiones,o puntos de control involucrados con la implementación del sistema, descripción de registros, decisiones sobre cómo se debe implementar el archivo, su modo de acceso, número y tipo de archivos intermedios a ser pasados entre programas, lenguajes de programación a ser utilizados en la implementación del sistema, etc..

La principal razón por la cual tal tipo de decisiones no deben ser incluidas en los requerimientos funcionales es que no tienen nada que hacer con la concepción que tiene el usuario de la aplicación. El usuario no debe preocuparse si el sistema será programado en *FORTRAN, COBOL*, o si va a utilizar una base de datos relacional o en

[26] Yourdon, E. "Structured Design"

red, simplemente debe esperar que el sistema produzca las debidas salidas con las respectivas restricciones de tiempo y dinero.

Se plantea que no sea el analista el que diseñe el sistema, pero sucede que en ciertas ocasiones, algunos analistas que anteriormente fueron programadores o diseñadores de sistemas, hacen algunas decisiones inconscientes mientras se entrevistan con el usuario, además, cuando se realiza el estudio de factibilidad, o el de costo-beneficio, motiva al analista a que en alguna parte del análisis, tome ciertas decisiones preliminares acerca del tamaño, poder y costo del equipo de cómputo requerido para resolver los problemas del usuario. Es por eso que se recomienda que ese tipo de decisiones sobre el diseño se hagan lo más tentativamente posible, de tal forma que pueda ser cambiado después si es necesario.

Por otro lado, el diseñador de sistemas es la persona que se encarga de definir el diseño estructural del sistema o de los programas, esto es, determinar cuáles son los subsistemas apropiados, programas o módulos y cómo están interconectados, diseñando los elementos de la base de datos, los componentes del sistema y las interfases entre ellos. El producto final del diseño es un documento que describe el diseño estructural.

Una vez que se ha terminado con el diseño, el programador espera recibir especificaciones sobre los módulos, que incluyan información referente a las entradas, salidas, interfases con otras partes del sistema, y los algoritmos mediante los cuales los módulos harán su trabajo.

II.6. METODOLOGIAS PARA DESARROLLAR SISTEMAS.

Una vez definido todo este proceso del ciclo de vida del desarrollo de sistemas, es necesario definir una estructura o un patrón de trabajo, basado en una metodología que nos permita desarrollar de mejor forma los sistemas. En la actualidad, existen muchas metodologías orientadas al análisis, diseño y programación de sistemas.

II.6.1. METODOLOGIAS CLASICAS

Las metodologías clásicas, están basadas en una descripción escrita del sistema, sin más herramienta que cierto tipo de cuestionarios, creando un gran texto que contiene toda la especificación funcional del sistema, platicada y en ocasiones incluye diagramas de flujo y/o algoritmos. En el diseño y la implementación, se utilizan diagramas de flujo, lenguajes tradicionales como lo son *COBOL, FORTRAN, BASIC*, etc.

Los problemas que representan este tipo de técnicas estriban principalmente en el hecho de que las especificaciones son escritas, generalmente son muy largas y en ocasiones, repetitivas, y por tanto se vuelve aburrido leerlas y difícil entenderlas.

Existen muchos problemas de comunicación entre usuarios y analista, lo cual se debe a la dificultad natural de describir un procedimiento, a las limitaciones de usar un texto narrativo, la falta de un lenguaje común, o bien a la ausencia de un modelo que nos represente el sistema. Además, debido a la naturaleza cambiante de los sistemas y a que la longitud y complejidad de la especificación hacen que sea muy difícil modificarse tan rápido como cambian los requerimientos. En consecuencia, las especificaciones rara vez son actualizadas para reflejar los cambios en los requerimientos. La falta de herramientas hacen que el proceso de análisis sea un tanto complejo y monótono.

Es debido a esto que muchas personas consideran que este proceso, no es realmente una metodología, y algunos han llegado a considerar esto como una situación caótica.

II.6.2. METODOLOGIAS ESTRUCTURADAS

Muchos de los problemas que generan fallas en la implementación del sistema se deben principalmente a un insuficiente análisis de requerimientos y especificaciones del diseño. Entre los años de 1960 y 1970, se desarrollaron muchas metodologías, principalmente las metodologías estructuradas[27], las cuales se crearon para imponer una estructura rígida dentro del ciclo de desarrollo de sistemas en las fases de análisis de requerimientos y en la especificación del diseño. Estas metodologías forman una disciplina tal que siguiendo la metodología, reducimos el riesgo causado por errores de requerimientos y de diseño.

Las principales metodologías están basadas en los trabajos realizados por Edward Yourdon y Tom De Marco (Análisis, Diseño y Programación Estructuradas), las cuales utilizan herramientas tales como Diagramas de Flujo de Datos, Diccionario de Datos, Diagramas de Estructura de Datos, para definir el análisis de requerimientos, Diagramas HIPO (Hierarchical Plus Input and Output) para definir la estructura de módulos, Diagramas Warnier-Orr y Diagramas de Entidad-Relación para modelar los datos etc Estas metodologías se aplican en los procesos de análisis de requerimientos y especificaciones del diseño[28].

Las técnicas estructuradas han dado buenos resultados y han sido aplicadas en muchos países En México, pocas son las compañías que desarrollan sistemas

[27] Fairley, L. "Ingeniería de Software"
[28] Fisher, A. "CASE, Using Software Development Tools"

utilizando este tipo de metodologías, de hecho pocas son las que siguen algún tipo de metodología.

Este trabajo tiene como objetivo el presentar algunas alternativas para poder desarrollar de mejor forma los sistemas, se hará referencia a las técnicas estructuradas, principalmente al diseño estructurado; en el siguiente capítulo se analizarán las características y se aplicarán a un caso práctico.

Podemos ejemplificar las metas de las técnicas estructuradas de la siguiente forma:

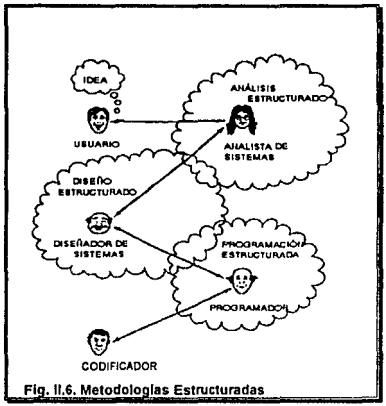


Fig. II.6. Metodologías Estructuradas

II.6.3. METODOLOGÍAS ORIENTADAS A OBJETOS

En los últimos años, se han desarrollado nuevas metodologías orientadas a objetos, las cuales están tomando auge con el desarrollo de la inteligencia artificial. Técnicas como el análisis, diseño e implementación orientados a objetos, la reingeniería

(Reescritura de código)[29], están tomando mucha fuerza, basicamente en las áreas de desarrollo de software. Algo que es particular de estas técnicas es que siguen basándose en las técnicas estructuradas, la diferencia estriba en la forma de manipular los datos, pero se siguen utilizando los diagramas de flujo de datos además de otras técnicas de diagramación.

En el capítulo 5 hablaremos más en detalle sobre esta metodología que aún se encuentra en su fase de desarrollo, lo cual no quiere decir que no se puede aplicar, sólo que falta darle una mejor estructura, lo cual está siendo posible gracias a las nuevas técnicas de **Ingeniería de Software Asistido por Computadora (CASE)** que incluyen ya este tipo de metodologías y se están popularizando rápidamente, principalmente los lenguajes orientados a objetos tales como C++, o PASCAL orientado a objetos, y retomando viejos lenguajes tales como ADA que han sido situados como aptos para realizar una eficiente programación orientada a objetos.

[29] Calderón, A. "Ingeniería de Software con la Metodología de Objetos"

CAPÍTULO III

"Las posibilidades están limitadas solamente por la imaginación y no por la Física".

NASA

CAPÍTULO III

LA INGENIERÍA DE SISTEMAS ... UNA ALTERNATIVA AL DESARROLLO DE SISTEMAS

La Ingeniería de Sistemas, está vinculada con la construcción de sistemas que suelen ser más grandes de lo que normalmente podría manejar un individuo. Esta técnica muestra una mejor estructura puesto que utiliza muchos de los principios de la ingeniería para el desarrollo de grandes sistemas. Además, se requiere de un profundo conocimiento de técnicas de computación, una facilidad de comunicación tanto oral como escrita y principalmente se debe estar bien familiarizado con el proceso de administración de proyectos relacionados con la producción de sistemas.

III.1. ADMINISTRACIÓN DE PROYECTOS

Cuando diseñamos sistemas, debemos tomar en cuenta todos los factores que intervienen en este proceso, tales como tiempo, recursos, elementos humanos, etc. Además es necesario saber como se deben administrar dichos factores y coordinar todas y cada una de las actividades que implica realizar un proyecto. La administración de proyectos facilita nuestras labores, minimiza la posibilidad de error y si es bien llevada, hace que el ambiente de trabajo sea ideal para los miembros del equipo.

Un proyecto puede ser definido como una actividad específica o como un grupo de actividades designadas a producir resultados específicos, con una limitada cantidad de tiempo, dinero y personas[30]. Con base en ésto, podemos distinguir ciertos elementos que pueden ser cuantificados o calificados y por tanto, medidos, Fig. III.1.

Fig. III.1. Administración de proyectos.

Dentro de la administración de proyectos existen tres fases, fig. III.2:

Fig. III.2. Fases de la administración de proyectos

[30] Hitachi, Co. "Introduction to Systems Design

En la fase de planeación se deben identificar las metas y las restricciones, definir las tareas, estimar las duraciones, asignar las tareas al personal, fijar los puntos de revisión, estimar los costos, identificar las fronteras, entradas y salidas y fijar el plan de actividades.

Cuando monitoreamos los avances, obtenemos una retroalimentación sobre si el progreso actual concuerda con los progresos planeados.

Para controlar el proyecto, se deben comunicar los progresos y los problemas del equipo, usuarios y directivos, y permitir que el plan se acomode a las circunstancias actuales para que se puedan lograr los objetivos.

Estas tres fases son iterativas y se repetirán en diferente medida hasta que el proyecto quede completo.

La construcción de grandes sistemas implica una gran cantidad de problemas, no se compara con los problemas que se pueden presentar cuando desarrollamos simples programas de computadora, cuyo grado de complejidad es tan simple que las personas pueden comprenderlos fácilmente y recordar todos los detalles del diseño, debido a que las especificaciones generalmente son informales y cuando realizamos una modificación, notamos de inmediato su efecto. Por el contrario, los sistemas grandes resultan ser complejos dado que son tantas las cosas que intervienen en su construcción que difícilmente se pueden recordar todos los detalles del proyecto. Es necesario implementar técnicas más formales de diseño, realizar una documentación adecuada para cada etapa del proyecto así como llevar una buena administración del mismo.

III.2. INGENIERÍA DE SISTEMAS

El concepto de Ingeniería de Sistemas se introdujo aproximadamente en 1960, gracias al desarrollo de la tercera generación de computadoras[31]. Debido a la gran capacidad de estas máquinas, era factible realizar aplicaciones que anteriormente eran irrealizables. La creación de dichas aplicaciones trajo como consecuencia el desarrollo de grandes sistemas de información.

En un principio, se utilizaron algunas metodologías que no fueron del todo adecuadas, se desarrollaban sistemas que tardaban mucho tiempo, a muy alto costo, poco confiables, difíciles de mantenerse y de muy bajo rendimiento[32].

---

[31] Sommerville, I. "Ingeniería de Software"
[32] Fujitsu, Co. "System Design Notes"

Se siguieron desarrollando nuevas técnicas y metodologías. Se piensa que el primer modelo de ciclo de vida del desarrollo de sistemas fue propuesto por primera vez por Royce en 1970[33], y desde entonces ese modelo ha presentado muchas variantes como las mostradas en el capítulo dos.

En forma general, todos estos modelos pueden resumirse en una lista de actividades tales como:

1. ANÁLISIS DE REQUERIMIENTOS

En donde se determinan los servicios, restricciones y objetivos del sistema que han sido establecidos por el usuario. Esto debe ser definido de una manera comprensible tanto para los usuarios como para el equipo de trabajo.

2. DISEÑO DEL SISTEMA

En esta fase se dará forma a las modificaciones propuestas, dividiendo los procesos en manuales y automatizados. Se pretende representar los procesos y las funciones del sistema a fin de posteriormente transformarlas en programas de computadora.

3. APLICACIÓN Y PRUEBA DE UNIDADES

Se pretende probar todas las unidades del sistema en forma individual, a fin de que cada una de ellas cumpla con su especificación.

4. PRUEBAS DEL SISTEMA

Todas las unidades del sistema se integran para realizar una prueba en conjunto. Después de dicha prueba, el sistema es entregado al usuario.

5. OPERACIÓN Y MANTENIMIENTO

Durante esta fase, el sistema instalado se pone en práctica, y se corrigen los errores que no se descubrieron anteriormente, además de mejorar los procedimientos y aumentar los servicios del sistema a medida que surgen nuevas necesidades.

Los puntos 3 y 4, generalmente se conocen como implementación.

Para este proyecto en particular, se ha seleccionado una metodología de ingeniería de software llamada ***"Metodología para la Ingeniería del Desarrollo de Software"*** *("Software Development Engineering Methodology")*, desarrollada por Fujitsu a finales de 1976, y sus herramientas de desarrollo fueron propuestas en 1977.

---

[33] Hitachi, Co. "Computer Technology in the '90s"

III.3. METODOLOGÍA PARA LA INGENIERÍA DEL DESARROLLO DE SOFTWARE

Esta metodología contiene varios aspectos importantes, pues utiliza el modelo de cascada visto en el capítulo II, para administrar el ciclo de vida del proyecto y además se basa en las técnicas estructuradas propuestas por Yourdon y DeMarco.

III.3.1. PRINCIPIOS BÁSICOS DE LA METODOLOGÍA

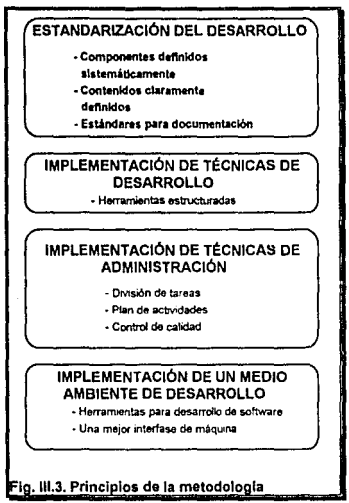


Fig. III.3. Principios de la metodología

a) La estandarizacion del proceso de desarrollo, supone que los componentes están sistemáticamente definidos, existe una clara indicación de los contenidos de cada componente y hay estándares para la documentación proveniente de cada proceso.

b) La implementación de técnicas de desarrollo nos provee de técnicas estructuradas para cada fase.

c) La implementación del medio ambiente para el desarrollo nos asiste con interfases de máquina tales como sistemas de tiempo compartido, un medio ambiente que incluye un editor generalizado de edición y administración de programas, generador de aplicaciones y administrador de sistemas, y un sistema de administración de la información en una base de datos relacional, además de la preparación de la documentación.

d) La implementación de técnicas de administración, nos provee de una guía para dividir tareas, hacer estimaciones, crear un plan de actividades, tener un control de calidad y una administración central para la revisión de datos.

Todo esto resulta de un enfoque de arriba hacia abajo, en donde los requerimientos del usuario son puestos en primer lugar después de realizar un amplio análisis y en seguida se realiza un diseño detallado en donde las necesidades de los usuarios son gradualmente cubiertas[34]. Esta metodología ha puesto algunas fases asociadas con algunos documentos y reportes que deben ser producidos. Después de cada fase, existen frecuentes puntos de revisión técnicos y de aceptación.

El proceso de administración del proyecto se ha integrado dentro del ciclo de vida del sistema[35]. En la actualidad, muchas empresas han incluido estas herramientas para el desarrollo de aplicaciones. **Fujitsu** ofrece una metodología y además un medio ambiente para el desarrollo. **NEC (Nipon Electric Company)** ofrece un medio ambiente semejante en sus equipos. Posteriormente veremos que dichas herramientas constituyen los principios de lo que hoy se conoce como herramientas **CASE (Computer Aided Software Engineering),** ingenieria de software asistida por computadora, que son herramientas muy útiles que facilitan el desarrollo de sistemas.

La metodología propuesta por Fujitsu, se describe en la figura III.4.:

---

[34] Fujitsu Co. "System design reference"
[35] Fujitsu Co. "System design notes"

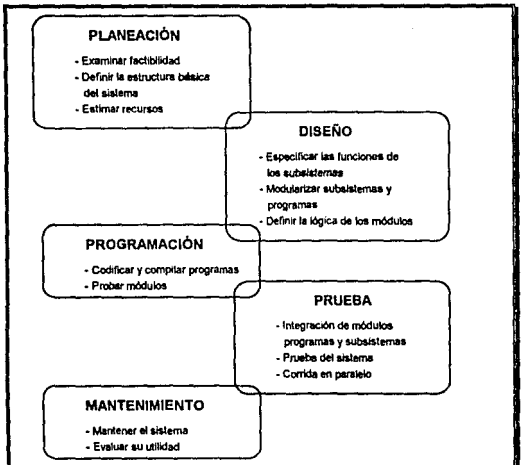


Fig. III.4. Descripción de la metodología

Planeación

En la fase de planeación, se pueden identificar 2 sub-fases:

1 Estudio y planeación

Se analizan las necesidades del usuario, se estudian sistemas similares y tendencias, se analiza el sistema actual, se definen las necesidades del usuario en términos de alcances y propósitos y se determina si es o no factible el continuar con el proyecto.

Cabe mencionar que una forma de realizar el estudio del sistema actual es con base en encuestas que se realizan por el equipo de trabajo. Además, los cuestionarios son una parte fundamental en la recopilación de la información, recogiendo opiniones, posturas, conductas y características de las personas que de alguna forma están involucradas en el funcionamiento de un sistema (los usuarios).

El proceso de recopilación de la información es muy serio, pues con base en esa información, se va a desarrollar un nuevo sistema y además, se debe tener mucho cuidado al realizar las entrevistas o los cuestionarios por que se pueden presentar problemas con los usuarios puesto que el planteamiento de preguntas tales como ¿cuáles son sus funciones?, ¿qué tipo de trabajo desempeña?, ¿qué problemas se presentan en su trabajo?, puede provocar malas interpretaciones por parte de los usuarios, en el sentido de que podrían pensar que se le esta revisando su trabajo o se esta viendo si trabaja bien o mal y peor aún, podría inventar la información para que no se piense que no hace bien su trabajo.

Es preciso que las preguntas se piensen muy bien antes de aplicarlas. Una pregunta mal hecha podría causar serios problemas de comunicación entre usuario y analista. Esto se debe principalmente a que no toda la gente esta dispuesta a que alguien revise su trabajo o bien no están dispuestos al cambio.

Existen diversos autores que han planteado la forma de efectuar entrevistas, cuestionarios y sobre aspectos psicológicos y de conducta de los seres humanos que pueden apoyar en el diseño de cuestionarios adecuados, algunos de ellos son:

Babbie, E. R., 1973.

Dillman, D. A., 1978.

Emory, C. W., 1985.

Kendall, K. E., J. R. Buffington, and J. E. Kendall, 1987.

Stephenson, W., 1953.

2 Planeación del Proyecto

Se realiza una planeación básica del proyecto en donde hacemos un diseño conceptual del sistema, se realiza un análisis de costo beneficio y se decide si se debe continuar basándose en los recursos y los planes de desarrollo.

Posteriormente, en la planeación detallada, se formulan las metas del producto, se especifica cuales son los recursos necesarios, se organiza al equipo del proyecto y finalmente se calendarizan las actividades del proyecto.

Después de cada fase, es necesario hacer una revisión del plan para determinar si se puede o no pasar a la siguiente fase.

Una vez terminada la fase de planeación, podemos entonces pasar al diseño y después a la implementación fig. III.5.

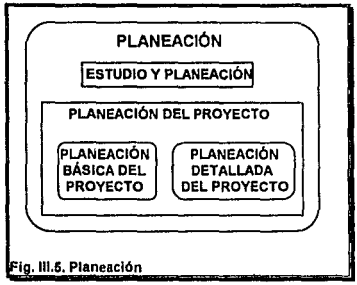


Fig. III.5. Planeación

Diseño

En la fase de diseño, encontramos dos sub-fases:

1. Diseño del sistema

Se realiza el diseño inicial del sistema definiendo sus funciones desde el punto de vista del usuario describiendo la configuración del sistema, y el diseño lógico, en donde se dividen las funciones de los subsistemas en programas y se enmarcan las interfases y relaciones entre los mismos.

El propósito de la revisión del diseño del sistema es checar que los subsistemas y el diseño de los programas sean consistentes y que cumplan con los requerimientos del usuario.

2. Diseño de programas

Se elabora el diseño de la estructura de los programas y luego de los módulos.

Con respecto a la estructura de los programas, se debe diseñar la lógica de los mismos, la cual debe ser consistente con todo el sistema, se debe definir la estructura de los archivos, especificar las condiciones de prueba, dividir los programas en

módulos y determinar las interfases entre ellos, definiendo finalmente su lógica. Nuevamente se realiza una revisión, esta vez correspondiente al diseño de los programas, la cual nos asegura que es congruente con el diseño global del sistema y que podemos programar con base en las especificaciones del diseño de los módulos.

Con ésto ya podemos empezar con la programación, en donde se codifican y compilan los programas, se prueban los módulos de acuerdo con las especificaciones de prueba y una vez terminado este proceso, se pasa a la fase de prueba fig. III.6.

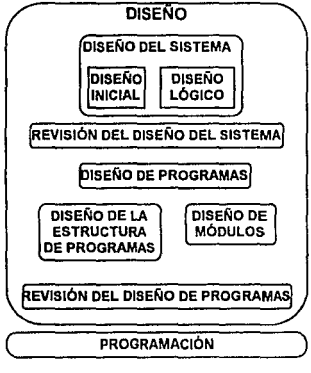


Fig. III.6. Diseño

Prueba y mantenimiento

En la fase de prueba se integran los módulos, programas y subsistemas, se checan las funciones contra las especificaciones para así realizar la prueba del sistema o prueba total, examinando las funciones, el comportamiento y las operaciones.

La revisión de la prueba se realiza para comparar el producto con el diseño del sistema. Esta es la revisión de aceptación en donde el usuario acepta el sistema y se verifica que la documentación sea compatible con el funcionamiento y las especificaciones.

En la prueba operacional, el sistema se pone en funcionamiento en paralelo con el sistema anterior para darle la oportunidad al usuario de que se acostumbre al nuevo finalizando con la destitución total del viejo sistema.

Cuando el sistema se encuentra en funcionamiento, empieza la tarea de mantenerlo y evaluarlo. Se debe mantener al sistema durante todo su período de vida, además es posible que se desee evaluar al proyecto y al sistema comparando lo estimado con lo actual, observando el desempeño y confiabilidad.

Una vez hecha la integración, las pruebas del sistema y las operacionales, concluimos con la fase de desarrollo. El mantenimiento seguirá todo el tiempo que dure el sistema fig III.7.

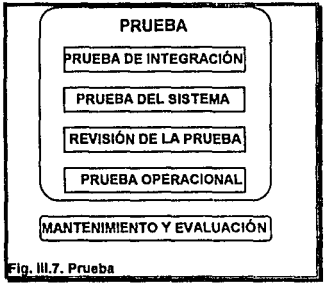


Fig. III.7. Prueba

A lo largo de todo este proceso, al final de cada fase, se generarán múltiples documentos que servirán de entrada para la siguiente fase. Esto beneficiará en gran medida por que nos asegurará una consistencia del sistema entre cada fase, además una serie de instrucciones claras en el manual de operaciones y en la guía del usuario nos asegurará que el sistema operará efectivamente en la fase de producción y nos facilitará el mantenimiento, y lo más importante es que nos brindará información estadística para otros proyectos.

Para tener una visión más objetiva de todo este proceso, se muestra a continuación un caso real en donde se aplica toda esta metodología. El ejemplo muestra únicamente el diseño del sistema por cuestiones de espacio y tiempo, ya que generalmente dentro del ciclo de vida del desarrollo de sistemas, la fase de análisis de requerimientos

ocupa aproximadamente entre el 65% y el 70% del tiempo total del proyecto. Además, el análisis es un proceso relativamente subjetivo y es más complicado hacer una ejemplificación del proceso. Es por eso que en este trabajo no se muestra todo el proceso de análisis y únicamente se dan los resultados y el documento final que es la especificación estructurada.

III.4. CASO PRACTICO

Kokuyo es una empresa japonesa que se dedica a fabricar todo tipo de artículos para oficinas, desde un lápiz hasta el diseño de oficinas completas. Los directivos de dicha empresa determinaron que requerían de información más oportuna de los empleados, así que pidieron al departamento de sistemas que realizara un estudio de factibilidad. Después de analizar los requerimientos y de efectuar un análisis de costo/beneficio, se propuso que la mayoría de las operaciones de administración del personal fueran convertidas a un sistema de base de datos en línea. El cálculo de salarios permanecería en Batch, porque sólo se realiza una vez por mes para todos los empleados.

El sistema propuesto tiene dos modos:

Sistema en Línea, el cual facilita la consulta, actualización y eliminación de registros de empleados. Esto se hace mediante el uso de:

- *una función de consulta para empleados actuales*
  datos personales
  datos históricos de sus puestos
  datos sobre su familia

- *una función que actualice los datos cuando un empleado cambia de departamento o de sección.*
- *una función que borre a los empleados cuando renuncien o se retiren de la empresa.*

Sistema en Batch

- *Listado de empleados que actualmente trabajan en un puesto*
- *Listado de empleados que previamente trabajaron en un puesto*

Con base en las especificaciones del sistema actual, se requiere diseñar e implementar un nuevo sistema.

La restriccion principal es que se cuenta con muy poco tiempo, pero se requiere de un

sistema bien diseñado que brinde información rápida, veraz y oportuna. Además se pide una documentación actualizada y completa, así como una base de datos eficiente y bien estructurada.

Entre los requerimientos se tiene que el sistema debe configurarse para una maquina **ACOS-4** de NEC con un sistema operativo multiusuario que lleva el mismo nombre, debido a que la empresa ya cuenta con ese equipo, además, la base de datos que se utilizará es la que ofrece **NEC** en el equipo, su nombre es **RIQS** (Relational Information Query System) sistema de información relacional,

Dentro del medio ambiente que presenta esta máquina se encuentra un administrador de aplicaciones llamado **VIS** (Versatile Information System), sistema de información versátil, que tiene como objetivo el permitir la creación de sistemas de bases de datos en línea en forma sencilla.

Este sistema permite implementar sistemas de bases de datos y de comunicación de datos tales como control de transmisión y recepción de mensajes, bases de datos compartidas o con control exclusivo, procesamiento de recuperación de fallas, y además provee funciones de soporte de desarrollo y operación de sistemas.

Dentro de los beneficios que se esperan obtener está el tener una respuesta inmediata de la información actualizada de los empleados, libre de redundancias, e información significativa en forma de listas.

CAPÍTULO IV

"Ningún problema es tan grande o tan complejo del cual no podamos escapar".

LINUS

CAPÍTULO IV

DISEÑO DEL SISTEMA

IV.1. MODELADO DE DATOS.

El análisis de datos es un proceso muy importante dentro del desarrollo de sistemas de información. Es gracias a este análisis como vamos a poder definir la estructura de nuestra base de datos, evitando así que existan fallas tales como falta de información, información redundante o poco significativa entre otras.

Cuando analizamos algún sistema, siempre vamos a encontrarnos con que existe cierto tipo de información que se almacena en diferentes archivos, ya sea en forma manual o en forma automatizada, pero siempre existirá cuando menos un archivo que contenga datos referentes al rol que juega dicho sistema.

Existen muchas razones por las cuales debemos prestar mucha atención en la forma de organizar nuestra información, ya que este recurso que puede ser acumulado, arreglado e integrado, corresponde a muchas de las necesidades de los usuarios.

En la actualidad, el soporte de las decisiones basadas en métodos científicos y matemáticos tales como las técnicas de investigación de operaciones se desenvuelven en áreas tales como la programación del desarrollo de productos, el control de la producción y el pronóstico de la demanda de algún producto.

Evidentemente es imposible pensar o hacer conjeturas sobre dichas cosas sin contar con datos actuales o pasados. Pronosticar la demanda de un producto es factible, siempre y cuando se provean datos tales como los resultados de ventas arreglados por producto y región.

Nuestra sociedad se ha convertido en una sociedad orientada hacia la información.

El objetivo principal del análisis de datos es el de estandarizar los datos que constituyen el sistema actual, clarificando sus relaciones, estructuras y características.

El diseño lógico de una base de datos es un problema muy difícil, especialmente cuando la base de datos es muy grande o muy compleja. Si una base de datos no se diseña correctamente, puede tener un serio impacto en las operaciones de la organización cuando esta sea utilizada.

A finales de los 70's y principios de los 80's, diversos investigadores han desarrollado herramientas para realizar la descripción de los datos y su interrelación con todo el sistema. Las principales son los diagramas de estructura de datos, los diagramas de entidad relación y los diagramas Warnier-Orr, los cuales se auxilian del diccionario de datos para realizar un modelaje efectivo de los datos.

El hacer una definición de la estructura de los datos efectiva, es una de las partes importantes del diseño de especificaciones dentro del ciclo de vida del sistema. La mayoría de las estructuras de los datos se deben diseñar antes de puntualizar todos los procesos o todos los módulos que contendrá el sistema. Una buena estructura de datos, siempre repercute en un buen diseño de procesos y de algoritmos, mientras que una mala estructura nos traería como consecuencia una excesiva cantidad de código redundante.

IV.1.1. DIAGRAMAS DE ENTIDAD-RELACIÓN

Los diagramas de Entidad-Relación (E-R) se usan para describir las relaciones entre los datos en una organización o en el modelo conceptual de un sistema o de un proceso. Estos diagramas son útiles para modelar la información que será almacenada en una base de datos.

Los datos son conjuntos de valores que representan las propiedades de una entidad. Las entidades se definen como un objeto o un grupo de objetos. Un objeto puede ser cualquier cosa, i.e., una persona, un empleado, un lugar, una organización o una función, acerca de la cual nosotros deseamos obtener o recolectar información fig. IV.1.

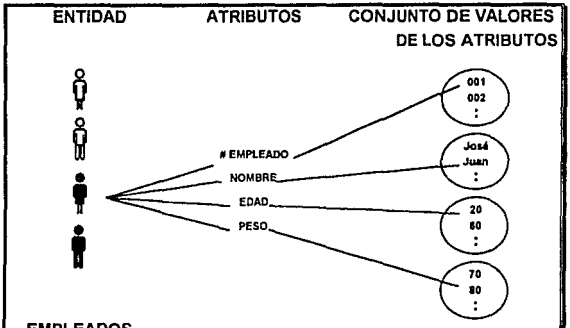


Fig. IV.1. Propiedades de las entidades

Las relaciones son asociaciones o interacciones que envuelven una o más entidades, nos muestran como una o varias entidades se relacionan con otra entidad o con otro grupo de entidades.

Los diagramas de E-R generalmente se componen de un rectángulo y de un rombo o diamante conectados por líneas rectas formando una red. Los rectángulos representan las entidades y los diamantes las relaciones. Algunas personas acostumbran incluir dos números entre entidades como se muestra en la figura IV.2.:

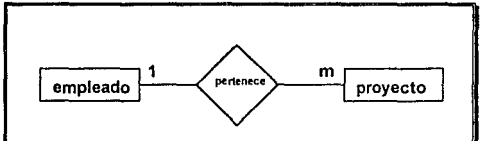


Fig. IV.2. Diagrama de Entidad-Relación

esto nos representa en qué proporción se realiza la relación, las más importantes son uno a muchos, muchos a uno, muchos a muchos. Por ejemplo si tenemos las entidades Empleados y Proyectos y la relación Pertenece, entonces podemos interpretar que un empleado pertenece a varios proyectos.

IV.1.2. DIAGRAMAS DE ESTRUCTURA DE DATOS

Nos muestra las relaciones entre archivos, auxiliándonos en la definición de la estructura de los mismos y para describir los requerimientos lógicos de la base de datos.

Este tipo de diagramas son útiles para algunas personas pero gracias al desarrollo de los diagramas de E-R su uso se ha disminuído, incluso se pueden usar para realizar los diagramas de E-R cuando aún no se está muy familiarizado con dicha técnica.

Se representan mediante círculos que muestran los archivos o los campos y una serie de flechas que indican la relación fig. IV.3.

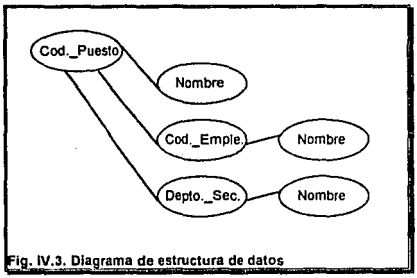


Fig. IV.3. Diagrama de estructura de datos

IV.1.3. DIAGRAMAS WARNIER-ORR

Esta es una de las técnicas de modelaje de datos más importante. Los diagramas Warnier fueron inventados por Jean-Dominique Warnier en Francia y posteriormente fueron mejorados por Kenneth Orr en los Estados Unidos de Norte América. Los diagramas Warnier-Orr son una simple técnica de representación de estructura de los

sistemas y se puede utilizar tanto para modelar los datos como para estructurar los módulos de un sistema. Aun cuando fueron creados para mostrar la arquitectura de programas, la mayor parte de las personas lo utilizan para describir la composición de las estructuras de datos.

El componente básico de dichos diagramas es un paréntesis. Se puede ver que los diagramas Warnier-Orr tienen la forma de un organigrama volteado hacia la derecha como se muestra en la figura IV.4.

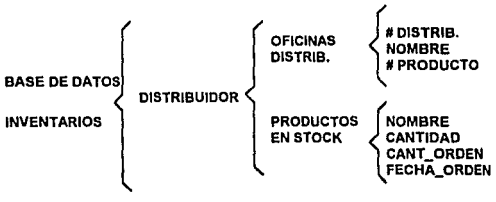


Fig. IV.4. Diagrama Warnier-Orr

El paréntesis más a la izquierda, nos representa el nivel superior, una vista global del sistema o de la estructura de datos. Sucesivamente, se pueden hacer vistas cada vez más detalladas o refinadas en forma jerárquica hacia la derecha; esto es que, la secuencia del refinamiento o de las actividades se presume sean de izquierda a derecha y de arriba a abajo. Estos diagramas muestran la composición de las estructuras, sin importar que éstas sean llamadas de subrutinas, definiciones de estructuras de datos o especificaciones sobre el formato de los archivos.

En base a la información proporcionada por estos diagramas, debemos definir una estructura que nos permita tener nuestros datos completamente optimizados. Esta estructura finalmente se convertirá en nuestra base de datos al rededor de la cual girarán todos nuestros procesos.

Todos los procedimientos que se ocupan para definir una estructura ideal de los datos, tiene su fundamento en la teoría de conjuntos y en el álgebra lineal. Para definir las relaciones entre datos, podemos utilizar la teoría de conjuntos:

Dados los conjuntos D1,D2,...,Dn, se desea crear un grupo que consiste de n elementos, un elemento de cada conjunto[16].

El conjunto R el cual contiene a este grupo como elemento es llamado relación, el conjunto Di es llamado Dominio, n es el grado, y el número de grupos en el conjunto R es el número cardinal fig. IV.5.

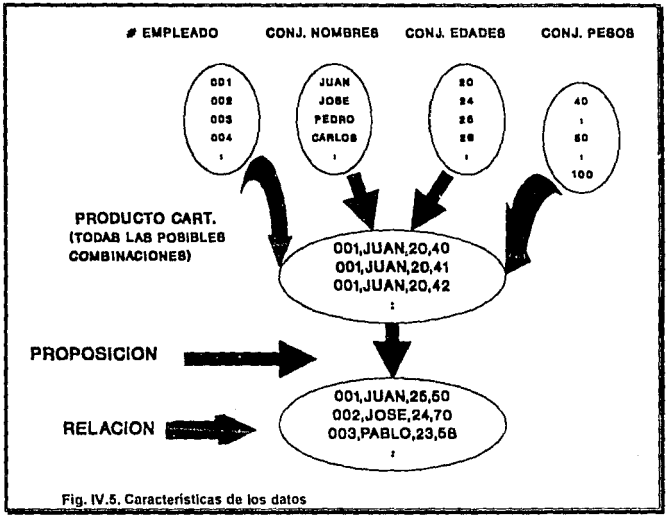


Fig. IV.5. Características de los datos

IV.2. NORMALIZACION DE ARCHIVOS

Una vez obtenidas las relaciones de datos es necesario considerar que desde el punto de vista del sistema completo. las relaciones entre datos debe ser optimizada para incrementar la eficiencia del almacenamiento y extracción de los datos. A este proceso se le conoce como normalización[17]

[16] Hitachi Co. "File design I"
[17] N.E.C. Co "Introduction to data bases"

Dado que las relaciones de los datos se obtienen básicamente para satisfacer las necesidades de los procesos, éstas no se encuentran optimizadas. Generalmente muestran las siguientes características:

- Los elementos que describen muchas entidades están mezclados en una relación simple

- Las relaciones dependientes entre los datos dentro de una relación son mezclados irregularmente.

- El mismo dato es duplicado en muchas relaciones de datos.

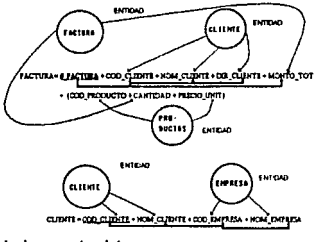


Fig. IV.6. Relaciones entre datos

Basándonos en la figura anterior, supongamos que deseamos hacer cierto mantenimiento de algunos datos, por decir, queremos cambiar la dirección del cliente, hacer un cambio en el precio por unidad o bien, como en nuestro ejemplo, queremos cambiar de puesto a un empleado, ¿cuántos serían los registros que se tendrían que modificar?.

IV.3. EL CONCEPTO DE NORMALIZACION

Dominio simple:

Un dominio simple es una área de definición que tiene un simple valor el cual no puede ser dividido.

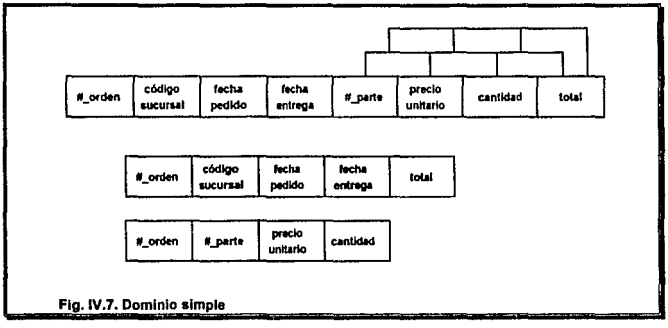


Fig. IV.7. Dominio simple

En este caso, sólo podemos tener para un número de orden, un solo código de distribuidor y una sola fecha de orden y entrega, pero podemos tener asociado a una orden, una serie de productos.

Dependencia funcional:

Dados dos dominios **DOM(A)** y **DOM(B)**, si la relación **(f)** entre los elementos de **a** que pertenecen al **DOM(A)** y **b** que pertenecen al **DOM(B)** en los dominios es tal que la determinación de un elemento depende únicamente de otro, entonces se dice que los elementos tienen una dependencia funcional[38].

La relación puede ser especificada como:

F:DOM(A) → DOM(B) o f:A → B

***Cuando esta relación se establece, se dice que B es funcionalmente dependiente de A*[39].**

[38] Hitachi Co. "Current System Study and Analysis"
[39] Hitachi Co. "File design II"

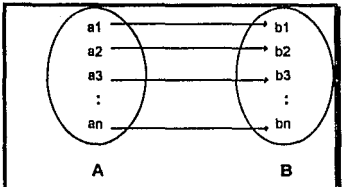


Fig. IV.8. Dependencia funcional

Existen tres tipos de dependencia funcional:

- **Dependencia funcional total**

En una relación, un elemento es dependiente solamente de todos los elementos llave.

- **Dependencia funcional parcial**

En una relación que tiene dos o más elementos llave, un elemento es dependiente de sólo algunas de esas llaves

- **Dependencia funcional transitiva**

Un elemento es dependiente solamente de elementos que no son llave.

Un elemento llave es aquel que nos sirve para accesar la información dentro de un dominio.

El proceso completo de normalización se muestra a continuación en la fig. IV.9:

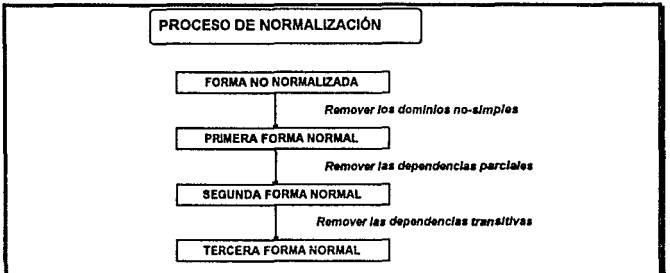


Fig. IV.9. Proceso de normalización

En nuestro ejemplo, el sistema original contaba con 4 archivos principales:

ARCHIVO DE EMPLEADOS
ARCHIVO DE SALARIOS
ARCHIVO HISTORIAL
ARCHIVO DE IMPUESTOS

Después del proceso de normalización se obtuvo la siguiente estructura de datos.

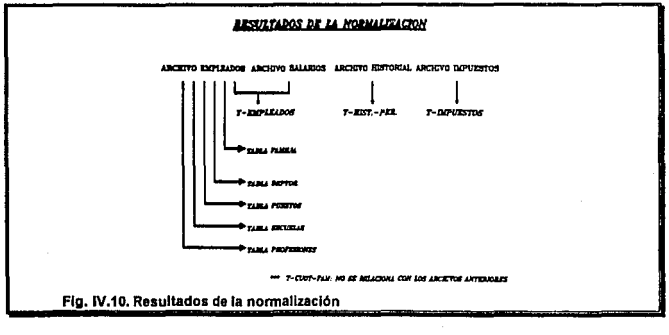


Fig. IV.10. Resultados de la normalización

IV.4. CREACION DE LA BASE DE DATOS

Una vez hecha la normalización, es necesario crear nuestra base de datos, para lo cual, se debe tomar en consideración los requerimientos especificados en el capítulo III.

Se indica que la base de datos debe ser del tipo relacional y debe ser implementada en una máquina ACOS-4.

Para mayor información sobre las bases de datos de tipo relacional sugerimos la sig. bibliografía:

Tasai 1990 "Sistemas de bases de datos".

NEC Corporation 1991 "Introduction to data bases"

En nuestro ejemplo, consideraremos que cada una de las dependencias funcionales obtenidas, formarán una tabla de la base de datos, es decir, tendremos un total de 9 tablas, las cuales contendrán la información de todo el sistema en forma óptima.

Para calcular el tamaño de la base de datos, debemos utilizar las especificaciones que dan los fabricantes de los equipos. En nuestro caso, dicha información se obtuvo de la guía del usuario para diseñar sistemas de administración de base de datos y del manual de instalación de la base de datos **RISQ** (relational information query system). Dicha base de datos requiere para su funcionamiento la creación de 3 archivos principales:

Archivos del directorio (RDIR)
Archivos de la base de datos (RDB)
Archivos de trabajo (RWK)

Una base de datos tipo RISQ consiste de un archivo RDIR y dos o mas archivos RDB y RWK.

Los archivos RDIR contienen información requerida para operar a la base de datos.

Los archivos RDB contienen datos de las tablas de los usuarios. Muchas tablas se pueden almacenar en un solo archivo RDB.

Los archivos RWK almacenan datos temporalmente durante el tiempo de proceso de la base de datos RISQ.

Los resultados se muestran a continuación.

BASE DE DATOS			
TAMAÑO DEL DIRECTORIO			
TABLA DE ADMON. DE USUARIOS			900 BYTES
TABLA DE ADMON DE OPERACIONES			1500 "
ARCHIVOS VSAS			8920 "
ARCHIVOS RIQS			600 "
TABLA DE ADMON. DE TABLAS			36450 "
TABLA DE ADMON. DE PROGRAMAS			56250 "
TABLA DE ADMON. DE AUTORIDADES			19000 "
		TOTAL	123620
124 Kb.	44 Blocks	4 Tracks	2 Cilindros

Los cálculos se realizan siguiendo las especificaciones técnicas de los equipos que se van a utilizar.

IV.5. DESCRIPCION DE PROCESOS

Una vez definida la estructura de los datos y habiendo determinado su capacidad, es necesario definir los procesos que se utilizaran para poder extraer la información de la base de datos.

Cabe mencionar que el diseño de sistemas (y en general todo el proceso de desarrollo de sistemas) no se aprenden fácilmente en un buen libro; el llegar a ser un buen diseñador de sistemas estriba en la experiencia y en el estudio de otros sistemas ya existentes, aunado con la implementación de técnicas y metodologías adecuadas. Un buen diseño es la clave de un buen sistema, el cual será fácil de aplicar y mantener, además de ser comprensible y confiable. Como ya hemos dicho, si un sistema esta mal diseñado puede funcionar bien, pero representa serios problemas en cuanto al mantenimiento y a su confiabilidad.

Muchos han sido los intentos por evitar un diseño defectuoso de sistemas, originalmente las especificaciones hechas en lenguaje natural, eran tomadas por el diseñador para hacer un diseño informal, el cual tenía la forma de un organigrama. De ahí se comenzaba con la codificación y el diseño se modificaba a medida que se

aplicaba el sistema[40]. Este procedimiento repercutía en serios problemas, puesto que al finalizar la etapa de aplicación, el diseño discernía por mucho de lo que las especificaciones iniciales marcaban convirtiéndose en una descripción totalmente inadecuada del sistema.

Debido a la gran cantidad de problemas generados por esta situación, se vió que el uso de notaciones completamente informales, como organigramas, son formas inadecuadas para formular y expresar el diseño de sistemas. Además se ha visto que una descripción precisa forma parte esencial de un buen diseño, y que el diseño de sistemas es una actividad repetitiva que contiene múltiples etapas las cuales no pueden representarse con un solo tipo de notación. Debido a esto, se han desarrollado varias notaciones para realizar el diseño de sistemas como los diagramas de flujo de datos, HIPO, PAD, SAD, de estructura, y lenguajes para la descripción del diseño.

Podemos definir 5 fases o etapas en el diseño:

1. **Deben establecerse los subsistemas que componen el sistema.**
2. **Cada subsistema debe dividirse en componentes individuales y ha de establecerse la especificación de los subsistemas, definiendo la operación de esos componentes.**
3. **Después cada programa se puede diseñar a base de subcomponentes que actúen recíprocamente.**
4. **Refinar cada componente.**
5. **Definir en detalle los algoritmos utilizados en cada componente.**

IV.4.1. DIAGRAMAS DE ESTRUCTURA.

Los diagramas de estructura nos auxilian describiendo un sistema como una jerarquía de partes mostrándolo gráficamente en forma de árbol[41]. Estos diagramas documentan la manera de aplicar los elementos de un diagrama de flujo de datos como una jerarquía de unidades de programa. Los diagramas fueron propuestos y modificados por Yourdon E., Constantine L. y Myers.

Dichos diagramas muestran las relaciones entre las unidades de programa sin incluir ninguna información acerca del orden de activación de esas unidades. Su simbología básica es:

---

[40] Sommerville. I "Ingeniería de Sistemas"
[41] Yourdon. E. "Structured Design"

1. **Un rectángulo con el nombre de la unidad**

2. **Una flecha que conecta los rectángulos**

3. **Una flecha con un círculo con el nombre de los datos que se pasan entre los elementos del diagrama de estructura. Las flechas con círculo se dibujan paralelas a las flechas que unen a los rectángulos del diagrama**

El diagrama de estructura correspondiente al ejemplo, se muestra a continuación:

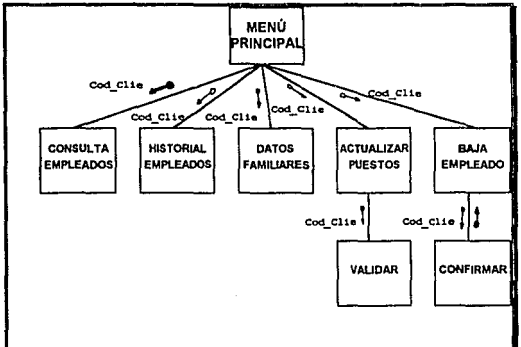


Fig. IV.11. Diagrama de estructura

CAPÍTULO V

"Un hombre debe saber sus limitaciones"

HARRY EL SUCIO

CAPÍTULO V

PERSPECTIVAS Y HERRAMIENTAS EN EL DESARROLLO DE SISTEMAS

Uno de los principales problemas a los que se han enfrentado las personas que desarrollan software, es el gran tamaño que algunos de estos sistemas llegan a tener, donde administrar el proyecto se convierte en una tarea muy difícil. Además, los avances tecnológicos han crecido en una forma sorprendente, esto es, hace tan solo 15 años, una máquina que tenía 64K de memoria principal, era considerada una computadora "grande", ya que "grandes" programas podían almacenarse en 64K[42]. Aún años después, con el desarrollo de las computadoras personales, se seguía considerando que 64K era el límite superior que un programa de aplicación grande podría consumir. Esto se debía un poco a que muchos de los programas de aplicación serios, eran hechos en lenguaje ensamblador, el cual proveía de la mejor ejecución en sus funciones y a su vez, limitaba el tamaño de los programas por la complejidad de su sintaxis.

No fue sino hasta 1981 cuando gracias a la gran aceptación que tuvieron las PC's en diversas áreas se rompió la barrera de los 64K's y se integraron las memorias de 640K, diez veces más grandes que la generación anterior. Se desarrollaron nuevos programas de aplicación, relativamente más grandes, pero con el reto de pasar el límite de tan vasta memoria. En la actualidad, la mayoría de los programas se realizan en lenguajes de alto nivel como lo son PASCAL, COBOL, C, etc., pero éstos son más complejos y sofisticados que antes. Ahora en muchos casos se requieren memorias superiores a 1MB, y los programas superan las 100,000 líneas de código.

Con programas de esta magnitud, es casi imposible lograr un 100% de confiabilidad, en cualquier momento es factible encontrar por lo menos un error, de hecho puede pasar mucho tiempo para depurarlo totalmente, sin contar los problemas inherentes al mantenimiento y actualización. Anteriormente, los pocos usuarios, dadas las

[42] Sommerville, I. "Ingeniería de Software"

circunstancias, podían tolerar una serie de errores en sus sistemas, pero difícilmente miles y miles de usuarios en la actualidad lo harán.

Muchos de los sistemas actuales han sido creados para empresas financieras como son los bancos, casas de bolsa, o bien, en el sector público. En general podemos hablar de dos tipos de mercado, aquel en el que los sistemas desarrollados son para una sola empresa (mercado horizontal) y aquel en el que un sistema se desarrolla para varios usuarios (mercado vertical).

Estudios realizados por la compañía TRW muestran que el 64% de los errores detectados, surgen de las fases de análisis y diseño, y sólo un 30% de estos errores son detectados antes de la prueba de aceptación, para que el sistema sea entregado al usuario.

En contraste, se sabe que sólo un 36% de los errores en los programas originados durante la fase de implementación fueron detectados, y el 75% de estos errores fueron detectados antes de la prueba de aceptación[43] fig. V.1.

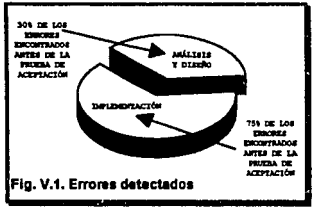


Fig. V.1. Errores detectados

Como ya hemos dicho, a lo largo de la historia han surgido muchas metodologías que nos ayudan a realizar de mejor forma los sistemas; las metodologías estructuradas, se han venido desarrollando e introduciendo desde hace aproximadamente 15 años. Estas metodologías proveen una estructura para el diseño y un conjunto de formalismos y prácticas que se han convertido en las bases para el desarrollo de sistemas complejos. Básicamente estas metodologías descomponen a un gran sistema en una red de pequeños módulos.

Históricamente se ha visto que el hombre, en su afán por crear un mundo mejor, ha desarrollado miles de productos que hacen su vida mas fácil. En la prehistoria, los hombres utilizaban piedras para contar sus pertenencias, después se creó el ábaco, Pascal y Leibniz desarrollaron calculadoras mecánicas que conformaron las bases

[43] Fisher, A. "CASE, Using Software Development Tools"

para la creación de las nuevas computadoras, todo con el fin de no perder el tiempo en labores repetitivas y en cálculos engorrosos, y aprovechar ese valioso tiempo en otras actividades importantes.

El desarrollo de sistemas no se queda fuera de este contexto; si bien es cierto que es un proceso que requiere de mucha imaginación y creatividad, es cierto también que hay actividades que son muy tardadas, repetitivas e incluso aburridas.

Dichas actividades pueden ser la generación de la documentación del sistema, la depuración de programas, la creación de los formatos de las pantallas o de los reportes, el dibujar los diagramas necesarios en el análisis y el diseño, y ¿por qué no? ... el programar.

Debemos tomar en consideración que el desarrollo de grandes sistemas trae como consecuencia tener que adoptar ciertas precauciones, administrar el proyecto, en fin, una gran cantidad de cosas que debemos tener en cuenta para no perdernos en la inmensidad del proceso.

En los últimos años, el hombre ha desarrollado una serie de herramientas que facilitan nuestra labor en el proceso de análisis, diseño e implementación, proporcionándonos más tiempo para encargarnos de asuntos más importantes como lo son la creatividad, la comunicación con los usuarios, e inclusive, evitar que el trabajo en el área de sistemas sea tan absorbente, puesto que en la actualidad, las personas que desarrollan sistemas, dedican mucho tiempo a su profesión y es muy común ver que muchas veces los programadores o cualquier miembro del equipo de trabajo se tiene que quedar "horas extras" para poder terminar el proyecto.

V.1. HERRAMIENTAS CASE

Las herramientas **CASE (Computer Aided Software Engineering)** Ingeniería de Sistemas Asistida por Computadora, reducen sustancialmente muchos de los problemas inherentes a los proyectos de desarrollo de sistemas de tamaño mediano a grande (aunque es muy útil también para los sistemas pequeños).

Cuando hablamos de herramientas **CASE**, podemos pensar que ayudan al analista en la misma forma que un **CAD-CAM** ayuda a un ingeniero. El ingeniero se apoya en esta herramienta para administrar su proyecto y para hacer un modelo o plano de la construcción que pretende realizar, y el analista utiliza los **CASE** para modelar, administrar, documentar, y crear un sistema.

Han sido ya varios los intentos por crear este tipo de herramientas, la siguiente figura muestra cómo han sido las transformaciones y qué se espera a futuro fig. V.2.

DÉCADA	CREACIÓN DE PROGRAMAS	PRUEBA DE PROGRAMAS	ANÁLISIS Y DISEÑO SISTEMAS
60's	COPYLIB. Surutinas y funciones integradas. Macros.	Generador de pruebas de datos.	
70's	Editores de código. Editores estructurados. Generación de código fuente a partir de un descriptor de macros.	Probadores y simuladores de funcionamiento.	Simuladores de desempeño. Definición y análisis de requerimientos, diseñadores de esquemas y prog. fuente para la conversión de diagramas.
80's	Generación de código a partir de una estructura ó diagrama gráfico. CASE para workstations Integración Host/WS	Monitoreo y covertura de pruebas. Monitoreo de errores acumulados. Probadores dinámicos	Soporte en la definición de pantallasy reportes, editores y analisadores de DFD's, normalización y modelaje de datos (editores de D-E-R).

Fig. V.2. Historia de las herramientas CASE

En la actualidad se han desarrollado una infinidad de herramientas **CASE**, tanto para equipos pequeños, como para grandes. Principalmente en los equipos grandes, las empresas que construyen este tipo de computadoras, desarrollan sus propios **CASE**. En el mundo de las computadoras personales, son varios los productos que han salido al mercado, cada uno de ellos con características similares. La diferencia estriba principalmente en la magnitud del paquete, y el tipo de metodologías que utiliza.

Existen muchas metodologías para desarrollar sistemas, pero la mayoría de ellas han surgido en torno a las estructuradas.

A continuación haremos un análisis de dichas metodologías, y veremos cómo nos pueden auxiliar las herramientas CASE cuando desarrollamos sistemas utilizando el análisis y el diseño estructurados u orientado a objetos.

V.1.1. METODOLOGÍAS ESTRUCTURADAS.

Como ya hemos dicho, el análisis estructurado fue desarrollado por Edward Yourdon y Tom DeMarco.

Esta metodología se enfoca principalmente en el flujo de datos de la aplicación (cómo fluye la información), en lugar del flujo de control, el cual es utilizado por las metodologías tradicionales o métodos de especificación de arriba hacia abajo (Top-Down). El objetivo del análisis estructurado es el crear un nuevo tipo de especificación: **La especificación estructurada**[44], la cual es un tipo de documento que describe los requerimientos funcionales. A diferencia de otras técnicas, la especificación estructurada muestra una variante que representa ciertas ventajas sobre las demás: es totalmente gráfica, lo cual difiere de las demás porque están completamente basadas en texto.

ESPECIFICACIÓN ESTRUCTURADA

Debido a que la especificación estructurada es gráfica, es factible crear una relación entre el analista que se enfoca en cómo construir sistemas y los usuarios que se interesan en saber qué es lo que se está construyendo.

El análisis estructurado, desde su inicio, ha utilizado 4 principales herramientas para desarrollar la especificación estructurada:

- **Los diagramas de flujo de datos**
- **El diccionario de datos**
- **Los diagramas de estructura de datos**
- **El español estructurado**
  (originalmente inglés estructurado)

basadas en 3 técnicas complementarias:

- **El flujo de datos**
- **El modelado de datos**
- **El control de especificaciones.**

En la actualidad existen muchas variantes de esta metodología que difieren en la simbología para realizar los diagramas e incluyen otro tipo de herramientas, pero la filosofía básica sigue siendo la misma.

---

[44] Yourdon, E. "System Design"

V.1.2. HERRAMIENTAS DEL ANÁLISIS ESTRUCTURADO

DIAGRAMAS DE FLUJO DE DATOS (DFD).

Un diagrama de flujo de datos es la representación en forma de red de un sistema (automatizado, manual o mixto), el cual declara las piezas componentes del sistemas y las interfases entre ellas.

Se usa principalmente para construir el modelo (físico y lógico) del sistema actual y del nuevo sistema propuesto, y para hacer una descripción funcional, produciendo finalmente la especificación estructurada.

Para describir el flujo de datos, es necesario definir todos los procesos; los sistemas complejos, contienen muchas funciones y procedimientos. Estas funciones o procedimientos, procesan los datos conforme pasan por el sistema. Los DFD's describen al sistema desde la perspectiva de los datos.

Las partes componentes de los diagramas de flujo de datos, básicamente son cuatro: El flujo de Datos, Los procesos, Las Fuentes externas y los Archivos y Bases de Datos.

Flujo de Datos: Estructuras de datos individuales que son transmitidas y recibidas por los procesos. Es decir, son conductos a través de los cuales fluyen los datos. Se representan principalmente por vectores, a los cuales se les asigna un nombre.

Notación:

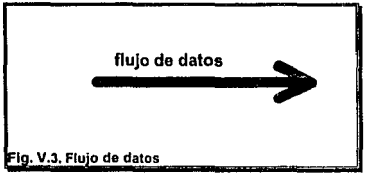


Fig. V.3. Flujo de datos

Procesos: Los flujos de datos fluyen hacia y desde los procesos. Los procesos son representados por círculos o burbujas, y transforman los flujos de datos de entrada en flujos de datos de salida.

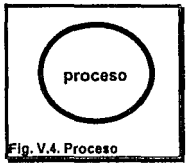


Fig. V.4. Proceso

Fuentes externas: Son personas, sistemas u organizaciones que están fuera del contexto de estudio, los cuales organizan, mandan y reciben flujos de datos que son parte de nuestro estudio.

Notación:

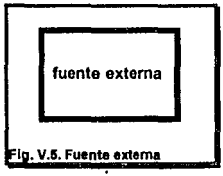


Fig. V.5. Fuente externa

Archivos y Bases de datos (almacenamiento de datos): Los procesos pueden requerir de archivos y de bases de datos para poder realizar sus operaciones. Los datos deben ser almacenados o sacados de los archivos, los cuales se representan con dos barras horizontales

Notación:

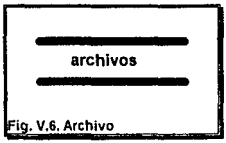


Fig. V.6. Archivo

Una de las características de los DFD's es que son diagramas jerárquicos. Cada proceso en un DFD, puede ser expandido o descompuesto en DFD's de niveles mas bajos. El primer diagrama que se presenta es el diagrama de contexto. Aquí se muestran todos los flujos de datos de entrada y de salida que va a tener todo el sistema y su conexión con las fuentes externas que rodean a nuestro sistema bajo estudio.
Después del diagrama de contexto siguen los diagramas de niveles intermedios, que son los que nos van a dar una descripción detallada de todo el sistema, partiendo de lo más general hasta lo mas particular, utilizando el principio de "Divide y Vencerás".

Otra característica de los DFD's es que son etiquetados para tener una mejor identificación de ellos y de sus respectivos descendientes, de tal forma que después del diagrama de contexto tendremos el diagrama cero, el cual contendrá una serie de procesos numerados (nota: la secuencia numérica no tiene nada que ver con la forma en que se procesan los datos, sólo son etiquetas que sirven para identificar los procesos), por ejemplo, si tenemos 4 procesos, a cada uno de ellos se les pondrá un número diferente del 1 al 4, y al proceso que se le asigne por ejemplo el número 1, originará otro diagrama al que se le llamará diagrama 1, y cada uno de los procesos que ahí se encuentren, tendrán los números 1.1, 1.2, etc.

Algo que es muy importante al dibujar DFD's, es la consistencia. Esto se refiere a que no debe perderse la relación de los flujos de entrada y salida a un proceso, con los de sus descendientes, por ejemplo, si al proceso uno entran el flujo A y el B, y sale el flujo C, cuando se dibuje el diagrama uno, deberán estar representados los flujos A, B y C.

En la siguiente gráfica se muestran los conceptos de consistencia y jerarquía.

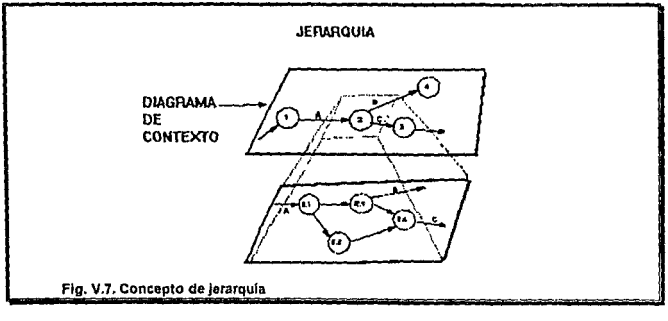


Fig. V.7. Concepto de jerarquía

DICCIONARIO DE DATOS

Los diccionarios de datos son catálogos o bases de datos de todos los elementos de dato encontrados en un DFD. Esto incluye los flujos de datos y los procesos. Todos los atributos de un dato en particular pueden ser encontrados en un diccionario de datos. Cada elemento de un flujo de dato es en realidad una estructura de dato. Los flujos de datos pueden ser tipos de datos sencillos como enteros, reales, cadenas; o bien pueden ser estructuras de datos mas complejas.

Su notación en muy sencilla:

símbolo	**significado**
=	**Está compuesto por**
+	**y**
[]	**seleccionar una**
{}	**iteraciones de**
()	**opcional**
**	**comentario**

Con estos símbolos, podemos describir todos los datos y los términos que usemos en nuestra especificación.

Por ejemplo:

Dirección_Cliente= { Calle + Número + (Número_Interior) + Ciudad + Estado + Código Postal)

Al igual que en los DFD's, es necesario checar la consistencia entre datos, además debemos evitar las redundancias, esto es que usando dos nombres diferentes, estemos describiendo a la misma estructura.

V.2. CASE Y LAS METODOLOGÍAS ESTRUCTURADAS

Cuando analizamos y diseñamos un sistema usando estas técnicas, las herramientas CASE facilitan nuestro trabajo cuando dibujamos los DFD's o cuando verificamos la consistencia de todo el sistema al igual que la consistencia en el diccionario de datos. En el capitulo IV se mostraron algunas otras técnicas estructuradas como lo son los

diagramas de estructura de datos, los diagramas de entidad-relación y los diagramas de estructura entre otros. Dichos diagramas fueron hechos con la ayuda de paquete (The System Architect) "El Arquitecto de Sistemas" el cual es una herramienta CASE para PC's que automatiza el proceso de generar y manipular diversos tipos de diagramas para el modelaje de sistemas como lo son:

- *Diagramas de Flujo de Datos de Gane & Sarson*
- *Diagramas de Flujo de Datos de DeMarco/Yourdon*
- *DFD's en tiempo real de Ward & Mellor*
- *Diagramas de estructura*
- *Diagramas de descomposición*
- *Diagramas de Entidad-Relación*
- *Diagramas de transición de estados*
- *Diagramas de transición de estados de Ward & Mellor*
- *Diagramas de flujo*
- *Diagramas para el diseño orientado a objetos*
- *Diagramas SSADM.*

Como vimos ya en el capítulo II, el ciclo de vida del desarrollo de sistemas en forma general comienza con el análisis de requerimientos, continúa con el diseño, la codificación, las pruebas y finalmente con la implementación y el mantenimiento. Hace algunos años, la ingeniería de sistemas asistida por computadora disponible daba como ya vimos herramientas de soporte para la implementación tales como lenguajes de programación de alto nivel, compiladores librerías. En la actualidad se le ha puesto mayor énfasis en modelar el sistema mucho antes de siquiera empezar la codificación, usando las metodologías de análisis y diseño estructurados o bien las recientes metodologías orientadas a objetos, que nos brindan una nueva alternativa y que se han desarrollado en forma muy acelerada.

V.3. CASE Y LAS METODOLOGIAS ORIENTADAS A OBJETOS

Las metodologías orientadas a objetos, son una nueva forma de pensar los problemas usando modelos que giran alrededor de conceptos del mundo real. El producto principal son los objetos, los cuales combinan la estructura de datos y su comportamiento dentro de una entidad. Los procesos y las funciones no se estudian y aplican en abstracto, sino sobre objetos reales, con propiedades específicas para producir sobre ellos objetos específicos[45].

[45] Calderón, "Ingeniería de Software con la Metodología de Objetos"

Básicamente el término "Orientado a Objetos" significa que vamos a organizar nuestros sistemas, principalmente el software, como una colección de objetos discretos que están formados por estructuras de datos y por comportamientos. Esto contrasta con la programación tradicional en el sentido de que las estructuras de datos y los comportamientos están vagamente ligados.

El enfoque orientado a objetos requiere de 4 aspectos principales:

1. *Identidad*
2. *Clasificación*
3. *Polimorfismo*
4. *Herencia.*

Características de los objetos.

IDENTIDAD

En una forma clara, la Identidad significa que los datos son cuantificados en entidades discretas y distinguibles llamadas objetos. Es la propiedad que distingue a cada objeto de los demás (incluyendo a los de su misma clase). Un ejemplo puede ser una párrafo en un documento, la cuenta de un cliente,una transacción bancaria, un caballo en un juego de ajedrez. Los objetos pueden ser concretos como lo es un archivo en un sistema o bien puede ser conceptual como una política de pagos en una empresa. Cada objeto tiene su propia identidad inherente. Esto quiere decir que aunque dos objetos tengan atributos idénticos (como nombre y tamaño) se consideran distintos[46], por ejemplo, si tenemos dos objetos "bicicleta", que tienen las mismas características, color y equipo, cada una tiene su propia identidad, y por tanto se consideran diferentes.

En el mundo real, los objetos simplemente existen, pero para un programa de computadora cada objeto tiene una etiqueta que permite que se le diferencie en forma única. Estas etiquetas pueden ser implementadas de diferentes formas, como direcciones, arreglos indexados, o bien como un valor único del atributo. Las referencias de los objetos son independientes de su contenido, permitiéndose colecciones mezcladas de objetos, como lo puede ser un directorio en un sistema de archivos, el cual contiene archivos y subdirectorios.

[46] Rumbaugh "Object Oriented Modeling and Design"

CLASIFICACIÓN

La clasificación significa que los objetos con la misma estructura de datos (atributos) son agrupados en clases. Perro, pieza de ajedrez, cuenta, son ejemplos de clases. Podemos decir que una clase es una abstracción que describe las propiedades que solamente son importantes para la aplicación e ignora el resto. Cualquier selección de una clase es arbitraria y depende exclusivamente de la aplicación.

Cada clase puede describir una posible cantidad infinita de objetos individuales. Se dice que cada objeto es una instancia de su clase. Cada instancia de una clase tiene sus propios valores para cada atributo, pero comparte los nombres de los atributos y las operaciones con otras instancias de la clase Fig. V.8.

POLIMORFISMO

Polimorfismo significa que las mismas operaciones se comportan de diferente forma en diferentes clases. La operación mover se comporta de diferente forma en las clases pieza de ajedrez y perro. Una operación es una acción o transformación que un objeto realiza o a la cual está sujeto. Mover, imprimir, pagar, son ejemplos de operaciones. A la implementación específica de una operación se le denomina método. Debido a que los operadores orientados a objetos son polimórficos, éstos pueden tener más de un método para ser implementados.

En el mundo real, una operación es una simple abstracción de un comportamiento análogo a través de los diferentes tipos de objetos. Cada objeto sabe cómo realizar sus propias operaciones. En los lenguajes de programación orientados a objetos, sin embargo, es precisamente el lenguaje el que automáticamente selecciona el método correcto para implementar una operación basándose en el nombre de la operación y la clase del objeto sobre el cual se está operando. Los usuarios deben saber cuantos métodos existen para implementar una operación polimórfica. Nuevas clases pueden ser incluidas sin necesidad de cambiar el código existente, Los métodos que ya han sido implementados pueden utilizarse para cada operación aplicable en las nuevas clases.

HERENCIA

La herencia se refiere a que se pueden compartir atributos y operaciones entre clases basándose en una relación jerárquica. Una clase puede ser definida en forma superficial y después se puede ir refinando en diferentes subclases. Cada subclase incorpora o hereda todas las propiedades de su superclase y agrega sus propiedades particulares. Un ejemplo puede ser una ecuación diferencial ordinaria y una ecuación

diferencial parcial. Las dos heredan las propiedades de la clase ecuaciones diferenciales, tales como que ambas incluyen derivadas. Otro ejemplo es una cuenta bancaria, la cual hereda sus propiedades a las subclases cuenta de ahorro, cuenta de cheques etc.

La habilidad para distinguir propiedades comunes dentro de un grupo de clases y asociarlas en una superclase, y el saber heredar las propiedades desde la superclase, puede reducirnos en forma considerable las repeticiones que suelen existir en el diseño y/o en los programas, siendo esto una de las principales ventajas de los sistemas orientados a objetos.

DESARROLLO DE SISTEMAS ORIENTADO A OBJETOS

Cuando queremos desarrollar un sistema usando metodologías orientadas a objetos, debemos pensar nuevamente en tener un ciclo de vida para este desarrollo, el cual incluye el análisis, diseño y la implementación. Nuestro interés principal en el desarrollo es el de identificar y organizar los conceptos en el dominio de la aplicación y no enfocarnos en su representación final en un lenguaje de programación orientado a objetos o no.

Se piensa que la parte difícil del desarrollo de sistemas está en la manipulación de su esencia debido a la complejidad heredada del problema, en vez de a los accidentes que se presentan cuando codificamos los algoritmos en algún lenguaje de programación en particular, lo cual se debe a imperfecciones temporales que pueden ser rápidamente corregidas.

El desarrollo de los lenguajes de programación orientados a objetos (LPOO) ha sido muy acelerado en los últimos años; es difícil encontrar libros sobre análisis y diseño orientados a objetos y esto repercute seriamente en la forma de administrar nuestro ciclo de vida de desarrollo de sistemas, nuevamente estamos cayendo en el enfoque "Slam Duck"; de ahí el interés de mostrar algunos aspectos importantes de la metodología.

El enfoque de desarrollo orientado a objetos, sugiere a los analistas y diseñadores, trabajar y pensar en términos del dominio de la aplicación a lo largo del ciclo de vida del desarrollo del sistema.

El principio básico de la metodología es el crear un modelo del dominio de la aplicación y después agregar los detalles de implementación durante el diseño del sistema.
Se contemplan las siguientes fases:

Análisis: Comenzando desde la descripción del problema, el analista construye un modelo de la situación mostrando sus propiedades importantes. El modelo obtenido es una abstracción concisa y precisa de lo que debe hacer el sistema deseado. Los objetos en el modelo deben ser conceptos del dominio de la aplicación y no conceptos computacionales tales como la estructura de datos.

Diseño del sistema: El diseñador de sistemas toma decisiones sobre la arquitectura global. Durante la fase de diseño, el objetivo del sistema se organiza en subsistemas basados en la estructura de los datos y en la arquitectura propuesta. El diseñador debe decidir sobre cuales características de operación debe optimizar, seleccionar una estrategia para atacar el problema y hacer un buen uso de los recursos.

Diseño de los objetos: Se debe construir un modelo del diseño basado en el modelo del análisis conteniendo los detalles de la implementación. En esta fase se debe prestar atención a las estructuras de datos y a los algoritmos que se necesitan para implementar cada clase. Las clases de objetos del análisis siguen siendo útiles, pero ahora se argumentan con estructuras de datos y algoritmos que han sido seleccionados para optimizar su comportamiento.

Implementación: Aquí las clases de objetos y las relaciones desarrolladas durante la fase de diseño de objetos son traducidas en un lenguaje de programación particular. La programación debe ser una parte mínima y mecánica del ciclo de desarrollo, debido a que las decisiones difíciles son realizadas en la fase de diseño.

Para poder describir al sistema se definen tres modelos diferentes[47].

Modelo de los objetos: Describe la estructura estática de los objetos y sus relaciones. Estos modelos utilizan diagramas de objetos, los cuales son gráficas cuyos nodos representan las clases de objetos y los arcos representan las relaciones entre clases[48].

[47] Rumbaugh "Object Oriented Modeling and Design"
[48] Calderón, A. "Ingeniería de Software con la Metodología de Objetos"

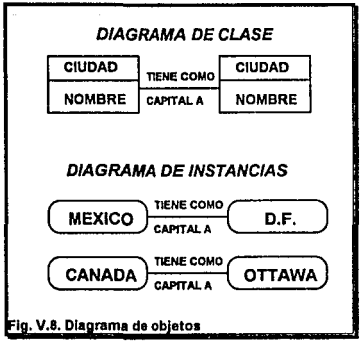


Fig. V.8. Diagrama de objetos

Modelo dinámico: Describe los aspectos del sistema que cambian a través del tiempo. Estos modelos se usan para especificar e implementar los aspectos de control del sistema. Este modelo contiene diagramas de estado, los cuales son gráficas cuyos nodos representan los estados y los arcos son las transiciones entre eventos causadas por los eventos.

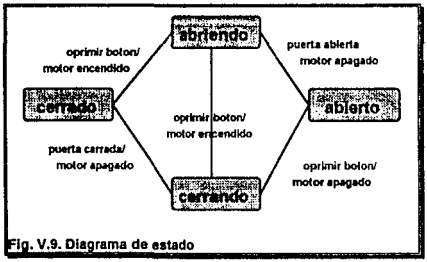


Fig. V.9. Diagrama de estado

Modelo funcional que nos describe los valores de transformación de los datos dentro del sistema. Este modelo funcional incluye diagramas de flujo de datos, cuyos nodos representan los procesos y los arcos el flujo de información.

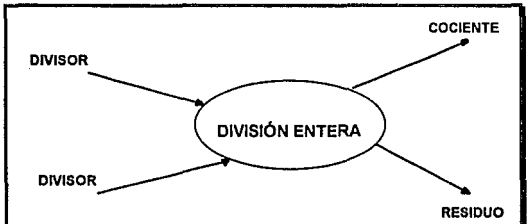


Fig. V.10. Diagrama de flujo de datos

En la actualidad no sólo el desarrollo de sistemas utiliza las técnicas orientadas a objetos, también la simulación ha comenzado a emplear dicha técnica, la simulación orientada a objetos ha dado grandes resultados, aunque aún se encuentra en su fase de desarrolo.

Existen algunas herramientas CASE que ya incluyen las metodologías orientadas a objetos y auxilian en el desarrollo de los modelos descritos anteriormente.

Como podemos observar las metodologías orientadas a objetos incluyen en cierta medida a las estructuradas, principalmente cuando se diseña el modelo funcional.

Algunas herramientas CASE no cuentan con la simbología o con una estructura propia para administrar un desarrollo que tenga un enfoque orientado a objetos, pero es factible utilizar los diagramas incluídos en el paquete, a fin de emular la diagramación y hacer una efectiva documentación.

El poder hacer una descripción más completa de esta metodología y las descritas por DeMarco y Yourdon, requieren de un tratamiento especial y quizá sería un solo tema de trabajo, es por eso que aquí sólo damos una breve descripción con el simple hecho de remarcar su importancia y cómo podemos usar ciertas herramientas que nos auxilien en el proceso.

CONCLUSIONES

Podemos concluir que el uso de metodologías auxiliares en el proceso de desarrollo de sistemas, es una práctica efectiva que permite lograr los objetivos planteados.

Muchas personas piensan que el utilizar una metodología determinada en el desarrollo de sistemas es un proceso difícil de realizar y probablemente sea más caro y más tardado que si solamente nos dedicamos a programar el sistema. Sin embargo, las experiencias adquiridas indican que el tiempo que nos tardamos en efectuar miles de cambios a los programas, se podría utilizar para realizar un buen análisis y diseño, dando como resultado que la elaboración de programas sea mucho mas sencilla y rápida.

Además, es fácil ver que cuando realizamos un análisis del sistema, podemos detectar muchas irregularidades en los procesos que se llevan a cabo en la empresa y es relativamente sencillo darse cuenta que áreas pueden ser consideradas como críticas o poco productivas.

Considero que el uso de ciertas metodologías y, mejor aun, la implementación de algún modelo de ciclo de vida como los analizados en este trabajo, es una

práctica que todos los profesionistas de la licenciatura en Matemáticas Aplicadas y Computación deben seguir, principalmente los egresados de la preespecialidad de simulación y análisis de decisiones, puesto que cuando hacemos el estudio del sistema actual, nos podemos dar cuenta de todas las operaciones que se realizan dentro de él, de las áreas a las cuales se les puede sacar más provecho y al momento de dar las alternativas de solución, es posible incluir cualquier tipo de modelo matemático que ayude a ser más productiva a la empresa.

BIBLIOGRAFÍA

Calderon Alzati, "Ingenieria de Software con la metodologia de Objetos"
APUNTES FUNDACION ARTURO ROSENBLUETH

Cárdenas, Miguel "La Ingenieria de Sistemas"
Limusa, 1974.

C.I.C.C. 1st. Batch, Group 1 "Personnel Management Information System Documentation"
C.I.C.C. Library, 1991

C.I.C.C. "System Design Workshop"
C.I.C.C. Library, 1991.

DeMarco Tom, "Structured Analysis and System Specification"
Yourdon Press, 1979.

Fairley, L. "Ingenieria de Software"
Mc. Grow Hill, 1989.

Fisher, Alan "CASE, Using Softwarw Development Tools"
Wiley, 1991.

Fitzgerald, Jerry "Fundamentals of Systems Analysis"
Wiley & Sons, 1981

Fujitsu Corporation "System Design Notes"
System Design Workshop, 1991

Fujitsu Corporation "System Design Reference"
C.I.C.C. System Design Workshop, 1991

Hitachi Corporation "An Overview of CASE"
Hitachi Computer Series, 1991

Hitachi Corporation "An Overview of OA System"
Hitachi Computer Series, 1991

Hitachi Corporation "Computer Technology in the 90's"
Hitachi Computer Series, 1991

Hitachi Corporation "Current System Study and Analysis"
Functional Analysis.
Hitachi Computer Series, 1991

Hitachi Corporation "Current System Study and Analysis"
Data Analysis
Hitachi Computer Series, 1991

Hitachi Corporation "Data Center Management"
Hitachi Computer Series, 1991

Hitachi Corporation "Detailed Design Methods I"
Hitachi Computer Series, 1991

Hitachi Corporation "Detailed Design Methods II"
Hitachi Computer Series, 1991

Hitachi Corporation "Program Design"
Hitachi Computer Series, 1991

Hitachi Corporation "Introduction to Systems Design"
Hitachi Computer Series, 1991

Kendall y Kendall "Análisis y Diseño de Sistemas"
Prentice Hall, 1991.

Lewis T. G., Smith B. J., "Computer Principles of Modeling and Simulation"
Houghton Mifflin, 1979.

Mora J. L., Molino E. "Introducción a la Informática"
Ed. Trillas, 1974.

NEC Corporation "Introduction to Data Bases"
Esfa Series, 1991.

NEC Corporation "Introduction to on Line Systemas"
Esfa series, 1991.

NEC Corporation "Program Creation Using SPD"
Esfa Series, 1991

Orilia Lawrence "Las Computadoras y la Información"
Mc. grow Hill, 1987.

Peters, Lawrence "Advanced Structured Analysis an Design"
Prentice Hall, 1987.

Pin-Shan, Peter "Applications of the Entity-Relationship Model"
Massachusetts Institute of Technology, Cambridge Mass., 1990.

Raumbaugh J., Blaha M. "Object Oriented Modeling and Design"
Prentice Hall, 1991.

Rodríguez, M. A. "Metodología de la Programación"
Mc. Grow Hill, 1991.

Senn, James "Análisis y Diseño de Sistemas de Información"
Mc. Grow Hill, 1992

Senn, James "Sistemas de Información para la Administración"
Gpo. Ed. Iberoamérica, 1990.

Sommerville, Ian "Ingeniería de Software"
Addison-Weslwy, 1988

Stuart, Ann "Writing and Analyzing Effective Computer System Documentation"
Holt-Saunders International, 1984

System Architect User Guide
Popkin Software & Systems Incorporate

System Architect Reference Manual
Popkin Software & Systems Incorporate

System Architect Tutorial
Popkin Software & Systems Incorporate

Tsai Alice, "Sistemas de Base de Datos"
Prentice Hall, 1990.

Uriegas, Carlos "Análisis Económico de Sistemas en la Ingeniería"
Ed. Limusa, 1987.

Weinberg, Victor "Structured Analysis"
Prentice Hall, 1988.

Yourdon, E., "Managing the Structured Techniques"
Yourdon Press, 1986.

Yourdon E., "Managing the System Life Circle"
Yourdon Press, 1981.

Yourdon E., "Modern Structured Analysis"
Yourdon Press, 1982.

Yourdon E., "Structured Analysis and System Specification"
Yourdon Inc., 1991.

Yourdon E., CONSTANTINE L. "Structured Design"
Yourdon Press, 1978.

"Sueña, sueña,
sueña hasta que
tus sueños se hagan
REALIDAD"

AEROSMITH