UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENIERIA EN COMPUTACION

Con estudios incorporados a la

Universidad Nacional Autónoma de México

" ESTUDIO DE LOS CRITERIOS DE CALIDAD DEL SOFTWARE "

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

JORGE BETANCOURT TEJEDA

Celaya, Gto. **Julio de 2001**

UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENIERIA EN COMPUTACION

Con estudios incorporados a la

Universidad Nacional Autónoma de México

" ESTUDIO DE LOS CRITERIOS DE CALIDAD DEL SOFTWARE "

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

JORGE BETANCOURT TEJEDA

Celaya, Gto. **Julio de 2001**

Este trabajo esta dedicado a todas y cada una de las personas que gracias a su apoyo hicieron posible la realización del mismo.

A mis padres:

Sr. Jorge Betancourt Romero y Sra. María Del Carmen Tejeda Rubalcava.

Ya que en todo momento han estado presentes para brindarme su apoyo, inculcándome el deseo de superación, enseñándome a seguir adelante sin importar los obstáculos difíciles que se presenten, ustedes que han sembrado amor, cariño, amistad y respeto; ha llegado el momento de cosechar, sinceramente gracias PAPÁ Y MAMÁ por darme tan buen ejemplo y hacer posible el logro de una meta mas y tengan siempre presente que los amo y admiro mucho, Gracias por creer en mi.

A mis Hermanos:

Mary y David.

Ustedes también sigan adelante en todo lo que se propongan recuerden que los triunfadores perseveran con lucha lacerante. Sabiendo que su valor no estriba en los triunfos que ha acumulado, sino en las veces que se ha levantado de sus fracasos, gracias por todas esas lindas sonrisas y travesuras que me han hecho muy feliz recuerden que los quiero mucho y cuentan conmigo por siempre.

A mis Abuelos y Abuelas :

Sr. Alberto Tejeda Palomares, Idolina Rubalcava Gonzalez y Maria de la Luz Romero Torres, por todos esos momentos, detalles y ejemplos tan bonitos que me han regalado, a ustedes que son el pilar de mi familia, gracias

A ti Verónica :

Por que en los momentos buenos y difíciles has estado a mi lado ofrecidome tu amor y cariño sin importarte nada mas ; muchas gracias por tu comprensión, por tu apoyo, por tu sinceridad y tu fidelidad a lo largo de todo este tiempo, me has hecho muy feliz ; En fin por ser como eres, nunca cambies, ya que en tus brazos descubrí el significado de la palabra "amor". Gracias te quiero mucho.

A mis Tíos y Tías:

José Alberto, Juan Manuel, Francisco Javier, Sergio Luis e Irma Idolina, por todos los consejos que me han dado, gracias por su apoyo.

A mis Amigos :

José Antonio Navarro, Luis Raúl Romero, Julio Nava, Martha Almanza, Araceli Jiménez H., Evencio López, Gabriella Ferrioli, Martha Nava, Ing. Carlos Alfonso Hernández e Ing. Arturo Alcaraz.

Gracias por todos esos buenos momentos en la Universidad, esto nos demuestra que la amistad es algo muy importante y que son de las cosas que jamas se olvidan.

También quiero mencionar a las personas que están o han estado presentes en algún momento, y que son una parte importante en mi vida, entre ellos a :

José Luis García Tavera, Angel Rangel, Juan Gabriel Ortega, Belem Magaña, Carmen Orozco, Juan Carlos Xicotencatl.

Especialmente al Ing. Fernando Villaseñor Velazquez.

Quien me ha ofrecido una amistad incondicional, orientándome y enriqueciéndome con sus consejos, depositando en mi la confianza necesaria para comenzar mi desarrollo de ingeniería a nivel profesional. Mi mas sincera gratitud para Usted.

Particularmente, deseo expresar mi agradecimiento a :

Lic. Araceli Lupercio, Ing. Laura Lizbeth Ramírez, Ing. Oscar Tellez, Ing. Miguel Angel Jamaica, Ing. Noé de Jesús Vela, Ing. Maya Gicela Villagómez, Ing. Anselmo Ramírez, quienes me entregaron sus conocimientos y me asesoraron para la elaboración de ésta tesis, y que gracias a su apoyo fue enriquecida con sus comentarios y sugerencias.

Finalmente, quiero dedicar este trabajo y agradecer a una persona muy importante para mi, la cual me apoyo en todo momento, enseñándome que mientras vivamos, podemos alcanzar muchas metas, y que la amistad se puede convertir en algo más fuerte, en algo **"especial".**

"Especial"
es una palabra, que describe algo único,
un abrazo, una puesta de sol
o alguien que ofrece amor
con una sonrisa o un gesto cariñoso.

"Especial"
describe a los que obran con el corazón
pensando siempre en los demás...

"Especial"
se aplica a algo precioso,
algo que admiramos
y que jamás podrá ser reemplazado...

"Especial"
es la palabra
que te describe mejor.

Con amor para :
Sr. Rogelio Betancourt Cuevas.
Gracias por haber sido "Especial".

Descanse en paz.

INDICE

INTRODUCCIÓN

Conforme a diversos estudios bibliográficos realizados, sobre los diversos métodos para implantar la calidad en el desarrollo del software, me percaté de algunas ventajas y desventajas que éstos ofrecen, las cuales pueden ser mejor comprendidas y optimizadas con lo que se expondrá a lo largo e esta tesis, haciendo un análisis crítico de los métodos existentes realizados de acuerdo con el estudio, lo cual se logrará profundizando en las ventajas de cada uno de los métodos y corrigiendo sus posibles desventajas.

El objetivo primordial de este trabajo es convencer a los ingenieros de software de los beneficios que se obtiene al utilizar el proceso administrativo, ya que de esta forma se asegura la calidad. La cual es un elemento actual que se está implementando en el desarrollo no sólo de software, sino en cualquier producto.

En el capítulo primero se presentan las características y componentes del software. Así mismo las de cualquier sistema de software, de igual manera los criterios de diseño del ingeniero de software, el capítulo segundo está dirigido a la organización y la importancia que tiene la estructura de equipos para el desarrollo de la misma, tomando en cuenta los elementos primordiales de toda organización y haciendo énfasis en la importancia que tiene la calidad en el mismo ; la planeación y los modelos de desarrollo del software, se tratan en el capítulo tercero, por último, en el capítulo cuarto se exponen las razones por las que se debe de tener un sistema eficaz de control, ya que con este aseguramos el camino al desarrollo de sistemas con calidad óptima apoyándonos en los principios y reglas de los sistemas de control.

CAPITULO I

SOFTWARE Y CALIDAD

Software- características- componentes- mitos del software- calidad- control de calidad- garantía de la calidad- aspectos de calidad que debe incluir el software

1.1 QUÉ ES EL SOFTWARE

En la década de los setenta muy pocas personas conocían la definición de software, debido a que no se encontraban en el ambiente de las computadoras, o quizá porque no existía la necesidad ni la revolución tecnológica de nuestra década. Ahora es más el porcentaje que podría dar una definición de lo que es el software, ya que la mayoría de nosotros estamos ambientados de una manera u otra con las computadoras.

Se entiende por software como un conjunto de elementos relacionados entre si los cuales se mencionan a continuación.

1) **Instrucciones** programas de computadora que cuando se ejecutan proporcionan la función y el rendimiento deseado

2) **Estructuras de datos** que permiten a los programas manipular adecuadamente la información

3) **Documentos** que describen la operación y el uso de programas.

4) **Instrucciones** aquellas responsables de que el hardware realice su tarea.[1]

1.2 CARACTERÍSTICAS DEL SOFTWARE

Es importante examinar las características del software para poder comprenderlo y diferenciarlo de otras cosas que el hombre puede construir. Cuando se construye hardware, el proceso creativo humano se traduce finalmente en una forma física; el software es un elemento del sistema que es lógico, en lugar de físico. Por tanto, el software tiene

[1] Véase en PRESSMAN, Roger S., ***Ingeniería del software,*** 4ª ed., McGraw-Hill, México, 1998, p. 7.

características notablemente distintas a las del hardware, las cuales se enumeran a continuación:

1. En un sentido clásico ***"el software se desarrolla, no se fabrica"*** Aunque existen ciertas similitudes entre el desarrollo de software y hardware, ambas son diferentes, en ambas se adquiere mediante un buen diseño; pero la fase de construcción de hardware puede introducir problemas de calidad que no existen o son fácilmente corregibles. En el software ambas actividades dependen de las personas, pero la relación entre las personas dedicadas a dichos trabajos son completamente diferentes. Para el software, las dos requieren la elaboración de un producto, pero los métodos son diferentes.

2. El software no se ***"estropea"***. Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen, suponiendo que no se introducen nuevos errores, podemos concluir algo: la idea de que el software no se estropea; pero si se deteriora, durante la vida de éste sufre cambios que son a consecuencia de su mantenimiento. Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, para corregirlos nuevamente, lentamente el nivel mínimo de fallos comienza de nuevo a crecer; es así como el software se va deteriorando debido a los cambios sufridos. Otro aspecto de este deterioro es que a diferencia de que en el hardware se estropee un componente, se sustituye por una pieza de repuesto. Para el software no existen piezas de repuesto. Cada uno de los fallos en éste indica un error en el diseño o en el proceso mediante el que se tradujo el diseño a código máquina ejecutable. Es por ello que el mantenimiento del software tiene una complejidad considerablemente mayor que la del mantenimiento del hardware.

3. ***La mayoría del software se construye a medida, en vez de ensamblar componentes ya existentes.*** Consideramos la forma en la que se diseña y se construye el hardware de control basado en un microprocesador; el ingeniero de diseño construye un sencillo esquema de circuitería digital, hace algún análisis

fundamental, asegurándose de que realice las funciones adecuadas y va al catálogo de ventas de componentes digitales ya existentes. Cada circuito integrado llamado "CI" o "Pastilla" tiene un número de pieza, una función definida y válida, una interfaz bien definida y un conjunto estándar de criterios de integración; después de seleccionar cada componente, puede solicitarse la compra. Por desgracia, los diseñadores de software no disponen de esa comodidad, no existen catálogos de componentes de software, se puede comprar software ya desarrollado, pero sólo como una unidad completa, no como componentes que pueden reensamblarse en nuevos programas.

4. ***La reutilización*** es una característica importante para un componente de software de alta calidad. Hoy en día, se ha extendido la visión de reutilización para abarcar no sólo los algoritmos, sino también las estructuras de datos. Los componentes reutilizables modernos encapsulan tanto datos como procesos que se aplican a los datos, permitiendo al ingeniero de software crear nuevas aplicaciones a partir de las partes reutilizables; por ejemplo, las interfaces interactivas de hoy se construyen frecuentemente a partir de componentes reutilizables que permiten la creación de ventanas gráficas, de menús emergentes y de una amplia variedad de mecanismos de interacción. Las estructuras de datos y los detalles de procesamiento requeridos para construir la interfaz están contenidos en una biblioteca de componentes reutilizables orientados a la construcción e interfaces.

1.3 COMPONENTES DEL SOFTWARE

Los componentes de software se construyen mediante un lenguaje de programación que tiene un vocabulario limitado, una gramática definida explícitamente y reglas bien formadas de sintaxis y semántica. En el nivel más bajo, el lenguaje refleja el conjunto de instrucciones del hardware; en el nivel medio, los lenguajes de programación se usan para crear una descripción procedimental del programa; en el nivel más alto, el lenguaje utiliza iconos gráficos u otra

simbología para representar los requisitos para una solución. Las instrucciones ejecutables se generan automáticamente.

Los lenguajes máquina son una representación simbólica del conjunto de instrucciones de la CPU. Si un buen programador produce programas de fácil mantenimiento y bien documentados, puede utilizar el lenguaje máquina para hacer un uso extremadamente eficiente de la memoria y para optimizar la velocidad de ejecución del programa. Si el programa está mal diseñado y tiene poca documentación, el lenguaje máquina puede convertirse en una pesadilla.

Los lenguajes de alto nivel permiten al programador y al programa independizarse de la máquina. Hoy en día existe una cantidad considerable de lenguajes de programación, poco más de una decena de lenguajes de programación de alto nivel con una aceptación buena en la industria.

1.4 APLICACIONES DEL SOFTWARE

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo). Excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales. El contenido y determinismo de la información son factores importantes a considerar para determinar la naturaleza de una aplicación de software. El contenido se refiere al significado y la forma de la información de entrada y de salida; el determinismo de la información se refiere a la predecibilidad del orden y del tiempo de llegada de los datos.

Un programa de ingeniería acepta datos que están en un orden predefinido, ejecuta el algoritmo sin interrupción y produce los datos resultantes en un informe o formato gráfico. Se dice que tales aplicaciones son determinadas.

Un sistema operativo multiusuario, por otra parte, acepta entradas que tienen un contenido variado y que se producen en instantes arbitrarios, ejecutan algoritmos que pueden ser interrumpidos por condiciones externas y produce una salida que depende de una función

del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Conforme aumenta la complejidad del software, es más difícil establecer compartimientos nítidamente separados.

A continuación se enlistan las áreas del software que muestran la amplitud de las aplicaciones potenciales:

- **Software de sistemas**

Es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (por ejemplo: compiladores, editores y utilidades de gestión de archivos) procesan estructuras de información complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo: ciertos componentes del sistema operativo, unidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área de software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

- **Software de tiempo real**

El software que mide, analiza y controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo y un componente de monitorización que coordina todos los demás componentes de forma que pueda mantenerse la respuesta en tiempo real (típicamente en el rango de un milisegundo a un minuto). Hay que tener en cuenta que el término tiempo real tiene un significado diferente de interactivo o tiempo compartido. Un sistema de tiempo real debe responder dentro de unas

ligaduras estrictas de tiempo. El tiempo de respuesta de un sistema interactivo (o de tiempo compartido) puede ser normalmente sobrepasado sin que se produzca ningún desastre.

- **Software de gestión**

El procesamiento de información comercial constituye la mayor de las áreas de aplicación del software. Los sistemas discretos (por ejemplo: nóminas, cuentas de haberes/débitos, inventarios, etcétera.) han evolucionado hacia el software de sistemas de información de gestión (SIG), que acceden a una o más bases de datos grandes que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas).

- **Software de ingeniería y científico**

Está caracterizado por los algoritmos de manejo de números. Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo, las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (del inglés CAD), la simulación de sistemas y otras aplicaciones interactivas, han comenzado a tomar características del software de tiempo real e incluso del software de sistemas.

- **Software empotrado**

Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de

control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicaciones en el salpicadero, sistemas de frenado, etcétera).

- **Software de computadoras personales**

El mercado del software de computadoras personales ha germinado en la pasada década. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, aplicaciones financieras, de negocios y personales, y redes o accesos a bases de datos externas, son algunos de los cientos de aplicaciones.

- **Software de inteligencia artificial**

El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Actualmente, el área más activa de la IA es la de los sistemas expertos, también llamados sistemas basados en el conocimiento; sin embargo, otras áreas de aplicación para el software de IA son reconocimiento de patrones (imágenes y voz), la prueba de teoremas y los juegos.

En los últimos años se ha desarrollado una nueva rama del software de IA llamada de redes neuronales artificiales. Una red neuronal simula la estructura de proceso del cerebro (las funciones de la neurona biológica) y a la larga puede llevar a una clase de software que puede reconocer patrones complejos y aprender de experiencias pasadas.

1.5 CALIDAD

El ***American Heritage Dictionary***[2] define la calidad como una característica o atributo de algo. Como un atributo de un artículo, la calidad se refiere a las características mensurables: cosas que se pueden comparar con estándares conocidos como longitud, color, propiedades

[2] En la obra citada en la nota 1, p. 122.

eléctricas, maleabilidad, etcétera; sin embargo, el software en su gran extensión, como entidad intelectual, es más difícil de caracterizar que los objetos físicos.

No obstante, sí existen las medidas de características de un programa. Entre estas propiedades se incluyen complejidad ciclomática, cohesión, número de puntos de función y líneas de código.

Cuando se examina un artículo según sus características mensurables, se pueden encontrar dos tipos de calidad: *calidad del diseño y calidad de concordancia.*

La calidad de diseño se refiere a las características que especifican los ingenieros de software para un artículo. El grado de materiales, tolerancias y especificaciones del rendimiento, todos contribuyen a la calidad del diseño. Cuando se utilizan materiales de alto grado y se especifican tolerancias más estrictas y niveles más altos de rendimiento, la calidad del diseño de un producto aumenta, sobretodo si el producto se fabrica de acuerdo con las especificaciones.

La calidad de concordancia es el grado de cumplimiento de las especificaciones de diseño durante su realización. Una vez más, cuanto mayor sea el grado de cumplimiento, más alto será el nivel de calidad de concordancia.

En el desarrollo del software, la calidad del diseño acompaña a los requisitos, especificaciones y al diseño del sistema. La calidad de concordancia es un aspecto centrado principalmente en la implantación. Si la implantación sigue el diseño, y el sistema resultante cumple los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta.

1.6 QUÉ ES EL CONTROL DE CALIDAD

El control de calidad es el proceso seguido por una empresa de negocios para asegurarse de que sus productos o servicios cumplen con los requisitos mínimos de calidad, establecidos por la propia empresa. Con la política de gestión (o administración) de calidad óptima (GCO) toda la organización y actividad de la empresa está sometida a un estricto control de calidad, ya sea de los procesos productivos como de los productos finales. La exigencia de una mayor

o menor calidad depende de muchos factores. Cuanto mayor es la vida del producto, menores serán las ventas, porque los consumidores no tendrán que volver a comprarlo, por lo que la calidad suele ser menor.

La importancia otorgada durante los últimos años al control de calidad es una respuesta a la competencia japonesa basada en la calidad. Sin embargo, fue un asesor económico estadounidense, W. Edwards Deming,[3] quien señaló que "el consumidor es la parte más importante de la línea productiva". Deming enseñó a los japoneses los distintos métodos de control de calidad.

Otro estadounidense, Joseph Juran, también desempeñó un papel crucial a la hora de promocionar la idea de vigilar la calidad y crear métodos de control. Entre los pasos que estableció para controlar la calidad destacan: la importancia de fomentar la idea de la necesidad de un control férreo de la calidad; la búsqueda de métodos de mejora; el establecimiento de objetivos de calidad y la aplicación de todo tipo de medidas y cambios para poder alcanzar estas metas; la necesidad de comprometer a los trabajadores en la obtención de una mayor calidad mediante programas de formación profesional, comunicación y aprendizaje, así como la revisión de los sistemas y procesos productivos para poder mantener el nivel de calidad alcanzado.

El control de calidad es una serie de inspecciones, revisiones y pruebas utilizados a lo largo del ciclo de desarrollo para asegurar que cada producto cumpla con los requisitos que le han sido asignados. El control de calidad incluye un ciclo de realimentación del proceso que creó el producto. La combinación de medición y realimentación permite afinar el proceso cuando los productos de trabajo creados fallan al cumplir sus especificaciones. Este enfoque ve al control de calidad como parte del proceso de fabricación.

Las actividades de control de calidad pueden ser manuales, completamente automáticas o una combinación de herramientas automáticas o interacción humana. Un concepto clave del control de calidad es que se hayan definido todos los productos y las especificaciones mensurables en las que se puedan comparar los resultados de cada proceso. El ciclo de realimentación es esencial para reducir los defectos producidos.

---

[3] Citado en STONER, James A. F., ***Administración,*** 6ª ed., Prentice-Hall, México, 1996, pp. 229-231.

1.7 GARANTÍA DE CALIDAD

La garantía de calidad o aseguramiento de la calidad consiste en la auditoría y las funciones de la información de la gestión; su objetivo es proporcionar la gestión para informar de los datos necesarios sobre la calidad del producto, por lo que se va adquiriendo una visión más profunda y segura de que la calidad del producto está cumpliendo con sus objetivos.

Por supuesto, si los datos proporcionados mediante la garantía de calidad identifican problemas, es responsabilidad de la gestión afrontar los problemas y aplicar los recursos necesarios para resolver aspectos de la calidad.

1.8 ASPECTOS DE CALIDAD QUE DEBE INCLUIR EL SOFTWARE[4]

La actividad de control de calidad del software está muy relacionada con las distintas actividades de comprobación y verificación que se efectúan en cada etapa del ciclo de vida del software. El control de calidad y otras actividades de verificación y comprobación son muy distintas; aunque algunas empresas no hacen diferencia entre éstas, las dos son muy distintas: el control de calidad es una función administrativa y la comprobación y verificación forman parte del proceso de desarrollo del software.

El control de calidad consiste en aquellos procedimientos, técnicas e instrumentos aplicados por profesionales para garantizar que un producto cumple o supera los estándares predefinidos durante el ciclo de desarrollo de un producto; sin estándares específicos preestablecidos, el control de calidad implica que un producto cumpla o supere un nivel mínimo industrial o comercialmente aceptable de calidad. Esto nos lleva a la conclusión de que es posible establecer estándares del software y, por otra parte, que se puede estimar el nivel de calidad de un producto del software.

Un estándar es determinada representación abstracta de un producto que define el nivel mínimo de rendimiento, robustez, organización, etcétera, que debe alcanzar el producto

[4] En apuntes de la materia ***Organización de centros de cómputo*** (octavo semestre).

desarrollado. La IEEE, ANSI y organizaciones militares se han dado a la tarea de desarrollar algunos estándares que describen los planes de administración de la documentación, prácticas de especificación, comparaciones de software, etcétera. Actualmente se elaboran estándares para la confiabilidad, las pruebas de desarrollo, etcétera.

El problema con los estándares es que tienden a ser muy generales, por lo que el control de calidad dentro de una organización requiere estándares más específicos.

A continuación se presenta el árbol de características de calidad de los productos de una forma clara y resumida, resaltando así las 7 características principales

- **Portable**.- esta característica hace referencia a la facilidad con la que un producto de programación puede ser transferido de un sistema de cómputo a otro o de un ambiente a otro.
- **Confiable**.- esta característica se basa en la capacidad de un programa de realizar una operación requerida bajo ciertas condiciones en un periodo de tiempo determinado.
- **Eficiente**.- grado con el que un producto de programación efectúa sus funciones, con un mínimo de recursos computacionales.
- **Ingeniería de Factores Humanos**.- esta característica es de las más importantes ya que refiere a la intervención necesaria del humano, gracias a estos podremos tener un software que sea amigable y estructurado que tenga comunicación con el usuario por medio de ayudas, además de ser accesible e integro.
- **Capacidad de ser probado**.- para esta característica se describe gracias a la estructuración del software, que sea capaz de ofrecer al usuario una comunicación continua y accesible para poder ser eficiente .
- **Entendible**.- de las características mas necesarias para cualquier software, su nombre denota la necesidad de que el software se presente de una forma clara y legible por el usuario además de ser consistente, estructurado y conciso.
- **Modificable**.- tomemos en cuenta que tarde o temprano el usuario final requerirá que al software se le hagan modificaciones por lo cual debe de ser capaz de soportar los incrementos necesarios en su estructura.

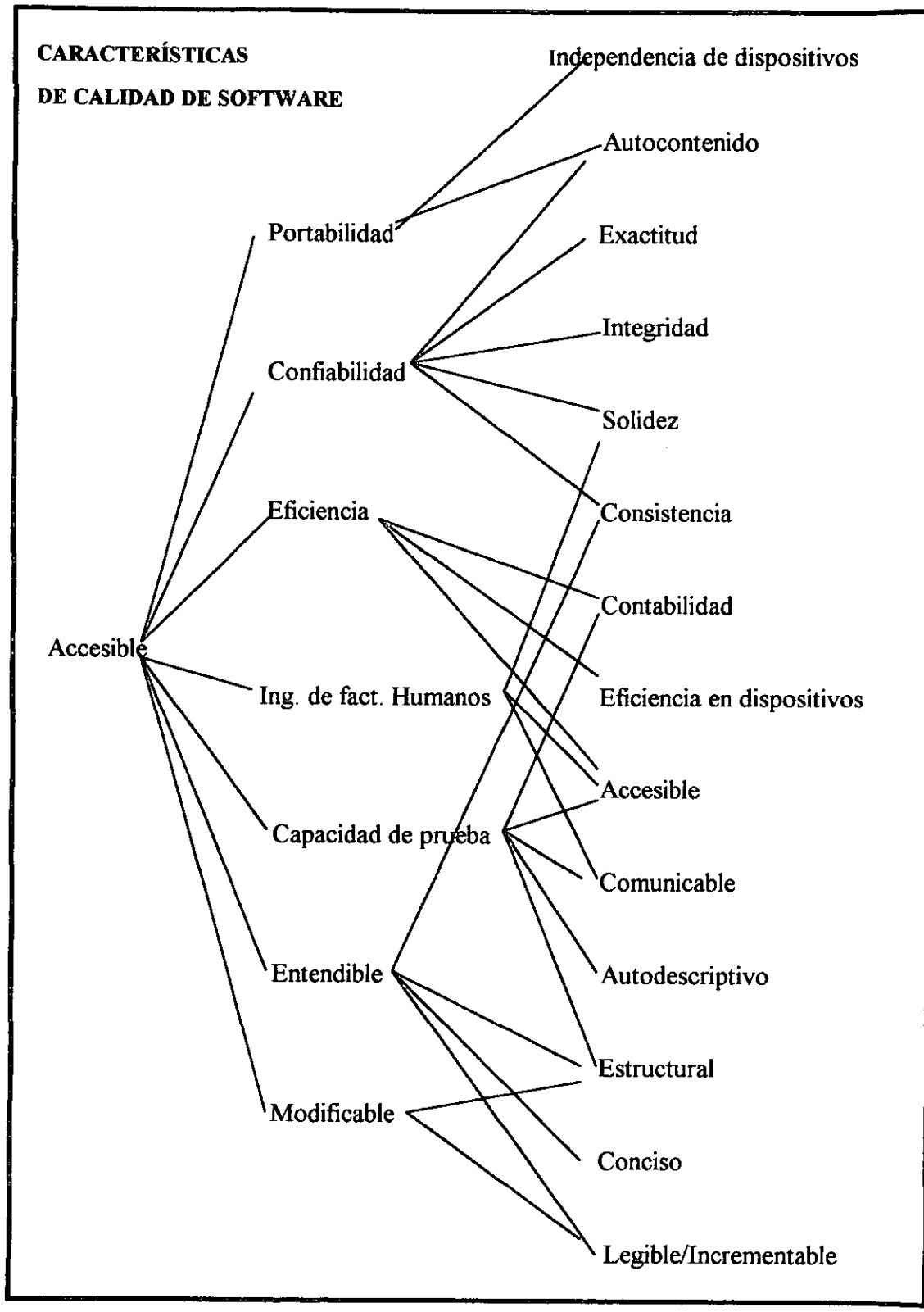
CARACTERÍSTICAS
DE CALIDAD DE SOFTWARE
Accesible
Portabilidad
Confiabilidad
Eficiencia
Ing. de fact. Humanos
Capacidad de prueba
Entendible
Modificable
Independencia de dispositivos
Autocontenido
Exactitud
Integridad
Solidez
Consistencia
Contabilidad
Eficiencia en dispositivos
Accesible
Comunicable
Autodescriptivo
Estructural
Conciso
Legible/Incrementable

FIGURA 1

Dentro de las organizaciones se recomienda que el control de calidad debe ser efectuado por un grupo especializado en el control de calidad del software independiente que dé sus informes directamente a la administración superior o a los encargados del proyecto.

El grupo de control de calidad no debe asociarse con ninguno de los grupos encargados del desarrollo del software, ya que es el responsable de supervisar que todos los grupos cuenten con los estándares de control de calidad que requiere el proyecto de la empresa.

Dentro de las actividades del control de calidad están las revisiones del diseño, recorridos del programa, etcétera., e informar sobre el control de calidad total del producto durante el periodo de desarrollo; implica también la revisión del producto terminado y que la documentación anexa se apegue a los estándares existentes. Este grupo podrá también estimar si las representaciones del producto son consistentes y completas.

1.9 CRITERIOS DE DISEÑO DEL PROGRAMADOR

Hoy en día, el programador debe tener criterios de diseño, los cuales lo harán diferente a los antiguos programadores. Anteriormente teníamos una idea falsa de lo que era el diseño de software, ya que solamente nos basábamos en lo que el cliente pedía, sin sugerir algo nuevo y novedoso. Esto nos hizo caer en un circulo vicioso, evitándonos observar con claridad que el diseñador debe de tener ciertos criterios dentro del desarrollo, los cuales nos aseguran el ciclo de vida de un sistema, el óptimo desarrollo del mismo, el manejo y lo mas importante, un software de calidad.

El ingeniero de software debe tomar en cuenta la naturaleza interactiva del proceso administrativo, tomando de él las ideas claras de los elementos que lo conforman, para poder así aplicarlos y obtener un diseño que respete los parámetros básicos de la calidad que debe de tener cualquier software.

El proceso administrativo advierte los siguientes elementos :

ORGANIZACIÓN

Es el proceso para ordenar y distribuir el trabajo, la autoridad y los recursos entre los miembros de una organización, con el fin de poder alcanzar las metas y objetivos antes marcados.

Existen diferentes metas, las cuales requieren diferentes estructuras. No es lo mismo realizar algún producto estandarizado que se elabora en serie en una línea de montaje, que la elaboración de un programa de software, ya que este requiere la formación de equipos de profesionales que deben interactuar con eficacia, no es posible organizarlos como si fueran trabajadores de una línea de producción.

PLANEACIÓN

La planeación implica presentar los objetivos de la organización y establecer procedimientos, idóneos para alcanzarlos. Además, los planes son importantes ya que con ellos obtenemos una guía para que:

- Cualquier organización obtenga y comprometa los recursos que se requieren para alcanzar sus objetivos.
- Para que los miembros de la organización desempeñen actividades congruentes con los objetivos y procedimientos elegidos
- El avance hacia los objetivos pueden ser controlados y medidos de tal manera que cuando no sea satisfactorio, se puedan tomar medidas correctivas.

Las relaciones y el tiempo son fundamentales para las actividades de la planeación. La cual produce una imagen de las circunstancias futuras deseables, dados los recursos actualmente disponibles, las experiencias pasadas, etcétera.

CONTROL

La función del control es asegurar los actos de los miembros de cualquier organización que, de hecho, la conducen hacia las metas establecidas.

El control entorna los siguientes elementos básicos:

1) Establecer estándares de desempeño.

2) Medir los resultados presentes.

3) Comparar estos resultados con las normas establecidas.

4) Tomar las medidas correctivas cuando se detectan desviaciones.

CAPITULO II

LA ORGANIZACIÓN

La organización- Concepto- Importancia- Principios de la organización- Sistemas de organización- Organigramas- Elementos de una organización- Estructura de equipos para desarrollo.

2.1 ORGANIZACIÓN

2.1.1 Concepto[5]

La palabra organización viene del griego "*organon*", que significa instrumento. Pero quizás ilustre mejor el significado de este concepto el uso que en nuestra lengua se da a la palabra "organismo"; éste implica, necesariamente:

a) **Partes y funciones diversas:** Ningún organismo tiene partes idénticas, ni de igual funcionamiento.

b) **Unidad funcional:** Esas partes diversas, con todo, tienen un fin común e idéntico.

c) **Coordinación:** Precisamente para lograr ese fin, cada una pone una acción distinta, pero complementaria de las demás; obran en vista del fin común y ayudan a las demás a construirse y ordenarse conforme a una teología específica.

La organización es la estructuración técnica de las relaciones que deben existir entre las funciones, niveles y actividades de los elementos materiales y humanos de un organismo social, con el fin de lograr su máxima eficiencia dentro de los planes y objetivos señalados.

La organización se refiere a estructurar; es quizás la parte más típica de los elementos que corresponden a la mecánica administrativa. Por lo mismo, se refiere a cómo deben ser las funciones, jerarquías y actividades. Por idéntica razón, se refiere siempre a funciones, niveles o actividades que están por estructurarse, más o menos remotamente: ve al futuro, inmediato

[5] Ver en REYES PONCE, Agustín, ***Administración de empresas,*** LIMUSA, México, 1993, p. 211; también en STONER, James, obra citada en la nota 4.

o remoto. La organización constituye el dato final del aspecto estático o de mecánica. Nos dice, en concreto, cómo y quién va hacer cada cosa (esto último, en el sentido de qué puesto, no precisamente de qué persona, y cómo lo va a hacer). Cuando la organización está terminada, sólo resta "actuar", integrando, dirigiendo y controlando, todo lo cual pertenece ya a la dinámica.

2.1.2 Importancia

La organización, por ser el elemento principal del aspecto teórico, recoge, complementa y lleva hasta sus últimos detalles todo lo que la previsión y la planeación han señalado respecto a cómo debe ser una empresa.

Es tan grande la importancia de la organización, que en algunas ocasiones nos ha hecho perder de vista que no es sino una parte de la administración, dando lugar a que la contrapongamos a esta última, como si la primera representara lo teórico y científico, y la segunda lo práctico y lo empírico.

Tiene también gran importancia por construir el punto de enlace entre los aspectos teóricos, que Urwick[6] llama de mecánica administrativa, y los aspectos prácticos, que se conocen bajo la denominación de dinámica; entre lo que debe de ser y lo que es. Al confundir la organización con la integración, presenta el peligro de mezclar la teoría con la práctica, lo ideal con lo real; con la consecuencia, muy frecuente, de que se pierdan de vista las metas.

Debemos precisar primero cómo debería ser nuestra organización, y después integrar ésta, como resulte más conveniente, tomando en cuenta los elementos de que disponemos, sin perder de vista a lo que debemos tender.

---

[6] Citado en REYES PONCE, Agustín, obra citada en la nota 7, pp. 212-213.

2.1.3 Principios de la organización

Dentro de la organización existen principios, los cuales nos aseguran el cumplimiento de los objetivos de la misma, el diseñador de software debe tomar en cuenta estos principios ya que gracias a estos lograremos cumplir los objetivos establecidos, apoyándonos en la especialización, unidad de mando y el equilibrio de autoridad-responsabilidad .

prefiero la palabra Principio a fin de evitar en lo posible la idea de rigidez, pues no hay nada rígido ni absoluto ; todo es cuestión de grado. Un mismo principio rara vez se aplica dos veces en la misma forma, por que debemos tener en cuenta las circunstancias diferentes y cambiantes en los seres humanos que son a la vez distintos y mutables, y por que además es preciso considerar otros elementos variables. También los principios son flexibles y pueden ser adaptados para atender cualquier necesidad ; todo es cuestión de saber como aplicarlos.

Es importante que cada desarrollador de software tenga conocimiento de los principios de la organización para la cual labora algún software, ya que estos deben estar considerados dentro del desarrollo.

A continuación se describen los siguientes principios :

- PRINCIPIO DE LA ESPECIALIZACIÓN

> *"Cuanto más se divide el trabajo, dedicando a cada empleado a una actividad más limitada y concreta, se obtiene, de suyo, mayor eficiencia, precisión y destreza."*

Este principio es fundamental en organización, ya que debe advertirse que la división del trabajo no es sino el medio para obtener una mayor especialización y precisión, profundidad de conocimientos, destreza y perfección en cada una de las personas dedicadas a cada función. Esto es el resultado natural de la limitación humana, ya que es imposible que mentes y capacidades privilegiadas abarquen todo; por ello, cuanto menor sea el campo al que se dedique una persona, obtienen más eficiencia en su trabajo.

- PRINCIPIO DE LA UNIDAD DE MANDO

> *"Para cada función debe existir un solo jefe."*

En este principio se establece la necesidad de que cada subordinado no reciba órdenes sobre una misma materia de dos personas distintas. Esto es fundamental para el orden y la eficiencia que exige la organización, ya que nadie puede servir a dos personas.

- PRINCIPIO DEL EQUILIBRIO DE AUTORIDAD-RESPONSABILIDAD

> *"Debe precisarse el grado de responsabilidad que corresponde al jefe de cada nivel jerárquico, estableciéndose al mismo tiempo la autoridad correspondiente a aquella."*

Es decir, que la autoridad se ejerce de arriba hacia abajo; la responsabilidad se ejerce en la misma línea, pero de abajo hacia arriba. Como elemento esencial en la jerarquía de una empresa, cada nivel jerárquico debe tener perfectamente señalado el grado de responsabilidad que en la función de la línea respectiva corresponda a cada jefe. Una autoridad sin responsabilidad trastornaría nuestra organización; es decir, una autoridad se delega, mientras una responsabilidad se comparte.

2.1.4 Sistemas de organización

Anteriormente se describieron los principios básicos de la organización, ahora debemos mencionar los sistemas organizacionales que se relacionan con dichos principios. Estos sistemas nos permitirán visualizar de una manera mas clara las líneas de responsabilidad y

autoridad que debemos establecer en toda organización. Estos sistemas son diversas combinaciones estables de la división de funciones y autoridades.

Existen tres sistemas fundamentales de organización: la organización lineal o militar, la organización funcional o de Taylor y la organización lineal y *staff*.

- Organización lineal o militar

Es aquella en que la autoridad y responsabilidad correlativas, se transmiten íntegramente por una sola línea para cada persona o grupo. (Ver figura 2.)

En este sistema cada individuo no tiene sino un solo jefe para todos los aspectos, ni recibe órdenes, consiguientemente, mas que de él, y a él sólo reporta. No nos parece correcto definir la organización lineal: "Aquella en que la autoridad y responsabilidad se transmiten " en línea", o "en línea recta", pues esto ocurre siempre, sino más bien: " por una sola línea", o "íntegramente para cada persona o grupo".

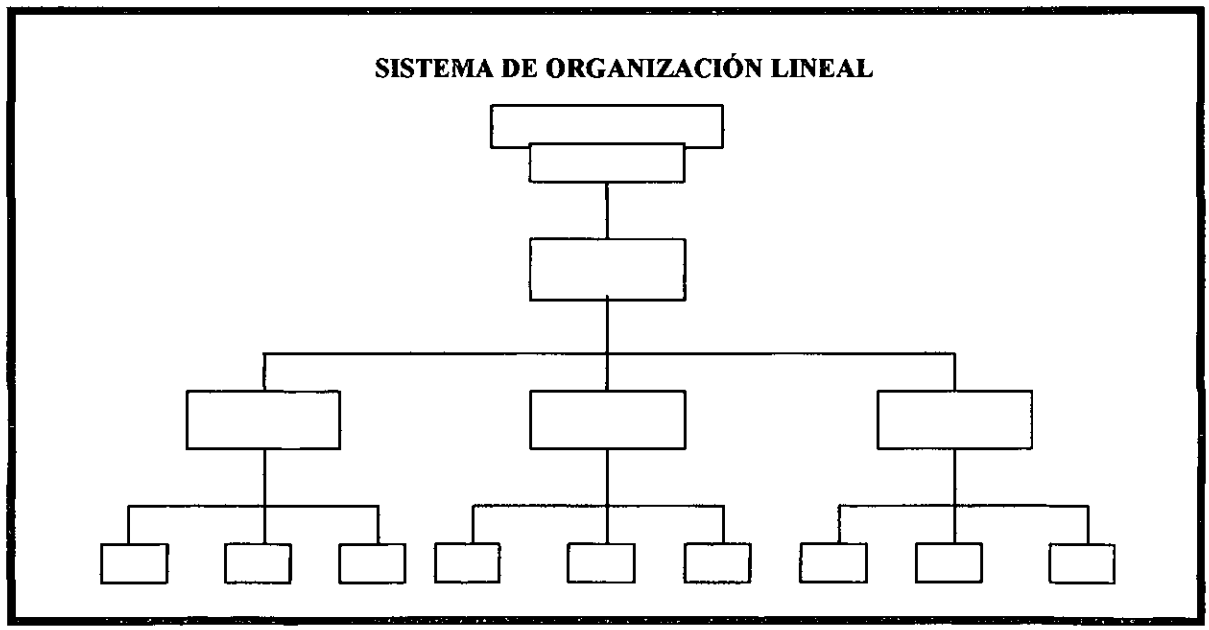

FIGURA 2[7]

Ventajas :

√ Es muy sencillo y claro.

[7] En REYES PONCE, Agustín, obra citada en la nota 7, p. 220.

√ No hay conflictos de autoridad ni fugas de responsabilidad.

√ Se facilita la rapidez de acción.

√ Se crea una firme disciplina, porque cada jefe adquiere toda su autoridad, ya que para sus subordinados es el único que la posee.

√ Es más fácil y útil en la pequeña empresa.

√ Obtenemos el beneficio de la especialización.

Desventajas :

- Es difícil capacitar a un jefe en todos los aspectos que debe coordinar.
- Se facilita la arbitrariedad, por que cada jefe tiene cierto sentido de "propiedad" de su puesto.
- Los jefes están siempre recargados de detalles.
- La organización descansa en "hombres", y al perderse uno de éstos, se producen ciertos trastornos.

- Organización funcional o de Taylor

La organización funcional es la forma más lógica y básica de la departa mentalización usada primordialmente por las pequeñas empresas. este tipo de estructura facilita mucho la supervisión además de facilitar el movimiento de habilidades para poder usarlas en los puntos donde mas se necesitan.

Taylor observando que en la organización lineal se da la "especialización", hacía notar que un mayordomo tenia conocimientos y responsabilidades de ocho campos (ver figura 3):

1) Tomar tiempos y determinar costos.

2) Hacer tarjetas de instrucción.

3) Establecer itinerarios de trabajo.

4) Vigilar la disciplina del taller.

5) Cuidar del abastecimiento oportuno de materiales, instrumental, etcétera.

6) Dar adiestramiento.

7) Llevar control de la calidad.

8) Cuidar del mantenimiento y la reparación.

Un mayordomo con estas capacidades no sería mayordomo, sino un jefe de rango superior.

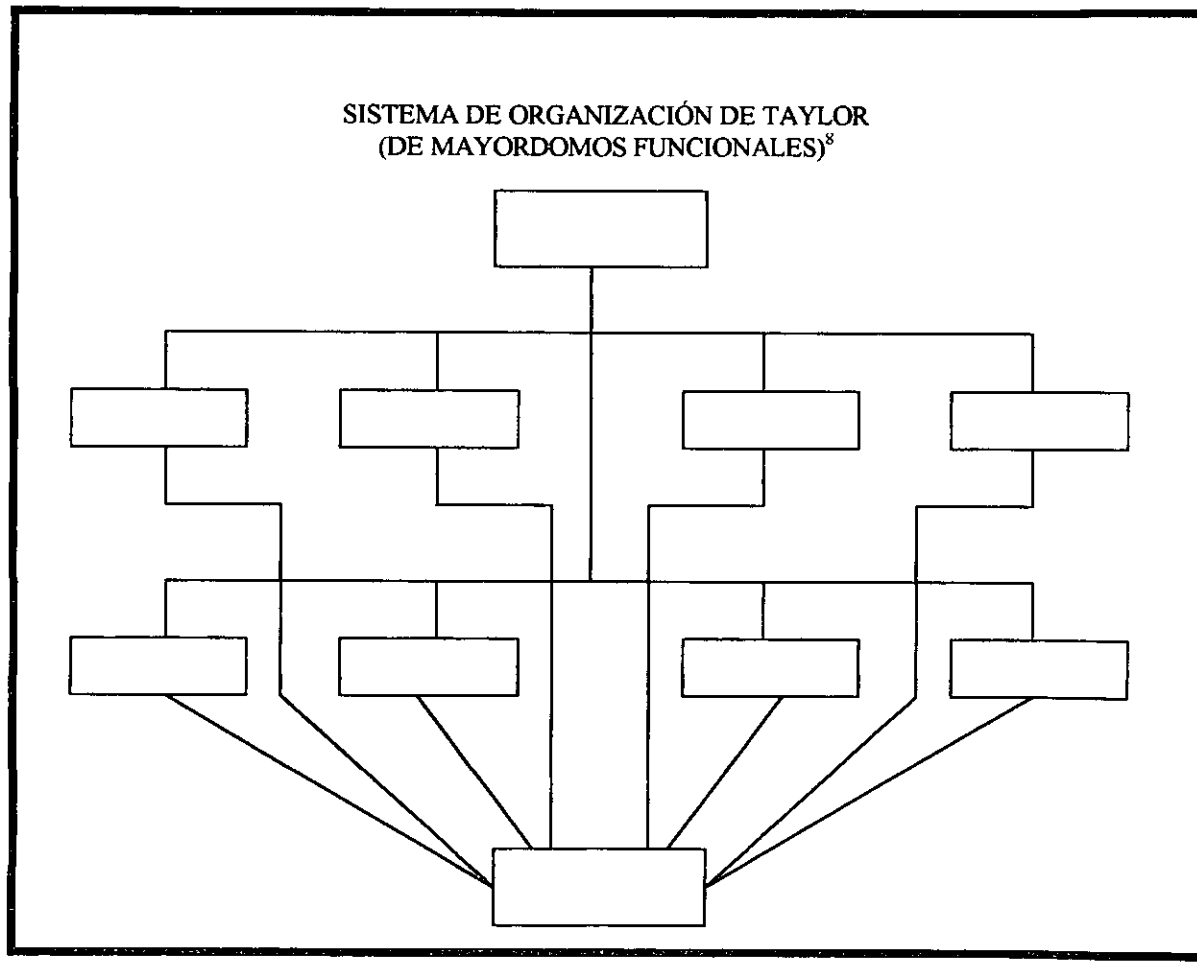

FIGURA 3

[8] Idem.

Para proveer remedio a tal situación, en este tipo de organización se propone que el trabajo del mayordomo se dividiera entre ocho especialistas, uno por cada actividad señalada, y que los ocho tuvieran autoridad, cada uno en su propio campo, sobre la totalidad del personal.

Actualmente, esto se realiza en los altos niveles de la administración. Un departamento de fabricación y un departamento de personal, ambos con autoridad en sus respectivas especialidades, sobre todo el personal respectivo. (Ver figura 4.)

SISTEMA DE ORGANIZACIÓN FUNCIONAL[9]

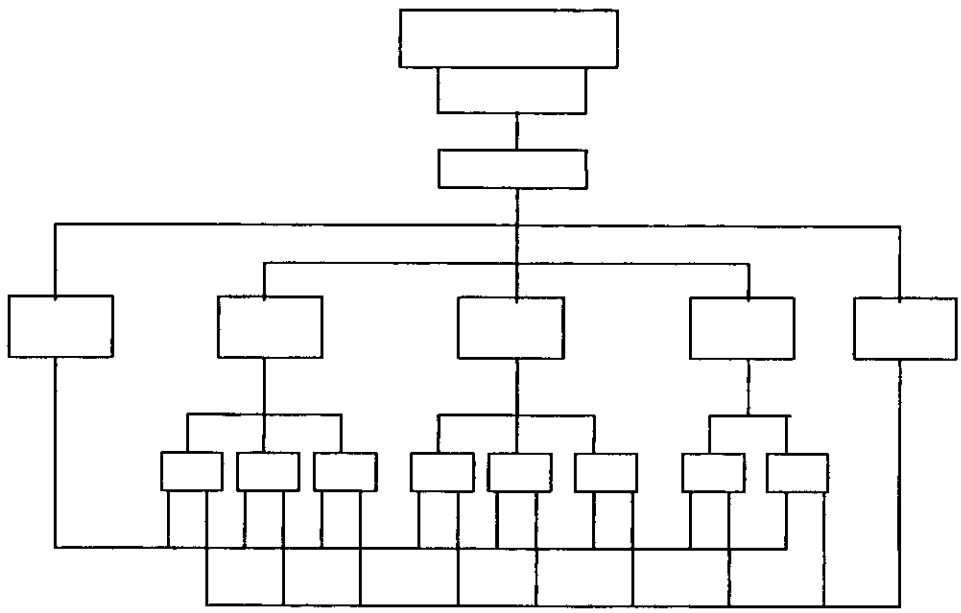

FIGURA 4

[9] Idem.

Ventajas:

√ Mayor capacidad de los jefes por razón de su especialización, y, por lo mismo mayor eficiencia.

√ Descomposición de un trabajo de dirección, complejo y difícil, en varios elementos más simples.

√ Posibilidades de rápida adaptación en casos de cambios de procesos.

Desventajas:

- Es muy difícil diferenciar y definir la autoridad y responsabilidad de cada jefe en los aspectos que son comunes a varios.
- Se da por ello con mucha frecuencia duplicidad de mando.
- Surgen por lo mismo fugas de responsabilidad.
- Se reduce la iniciativa para acciones comunes.
- Existen, fácilmente, quebrantamientos de la disciplina y numerosos conflictos.

De hecho, donde se da este sistema, un departamento suele predominar sobre los demás, originando, en cierto modo, el sistema que se presenta a continuación, aunque con graves defectos.

- Organización lineal y *staff* (lineal y de estados mayores)

Este sistema trata de aprovechar las ventajas y evitar las desventajas de los dos sistemas anteriores explicados (ver figura 5):

a) De la organización lineal conserva *la autoridad y responsabilidad* íntegramente transmitida a través de un solo jefe para cada función.
b) De la organización funcional conserva la forma básica y lógica de la departamentalización permitiendo a la vez la supervisión de los mismos.

Este sistema es el más usado actualmente por las grandes organizaciones, ya que todo el secreto de su éxito depende de precisar "el asesoramiento y servicio".

Un cuerpo asesora cuando se investiga permanentemente que puede mejorarse o innovarse, planeando esas nuevas mejoras para su empresa en concreto, sugiriendo los planes a la gerencia, hasta obtener su plena aprobación, obteniendo la aceptación y colaboración de los jefes de línea, a base de convencimiento; instruyendo la implantación de los nuevos sistemas, ayudando a establecerlos, resolviendo cualquier duda o problema que se puedan presentar en su operación, sobre todo al principio, para después revisar permanentemente los resultados, para estar siempre en condiciones de hacer sugerencias de mejora.

SISTEMA DE ORGANIZACIÓN LINEAL Y STAFF [10]

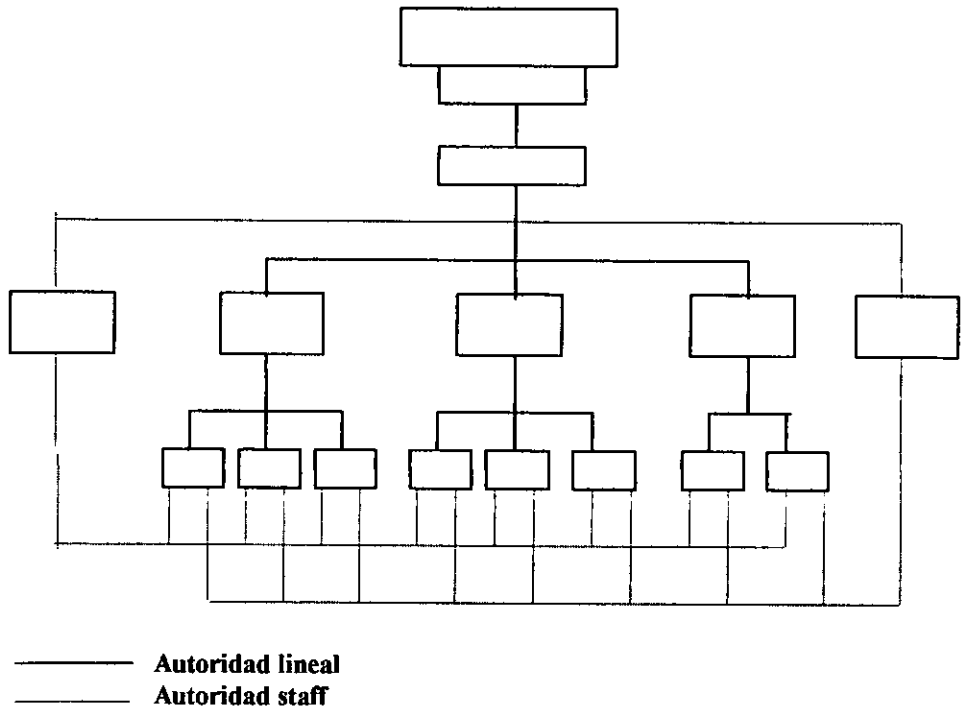

FIGURA 5

[10] Idem.

Un cuerpo sirve cuando realiza tareas en nombre de los jefes de línea y lleva a cabo ciertas funciones en representación de los mismos. Estos servicios se fundan ordinariamente en alguna de las siguientes razones:

a) Los jefes de línea no tienen la preparación específica que requiere la eficiencia del servicio que ellos deberían realizar, como las técnicas de selección de personal, de organización, de control estadístico, etcétera.

b) Los jefes de línea, aun suponiendo que tengan dichos conocimientos, no tienen tiempo de realizar esas funciones por sí mismos.

c) Por razones de uniformidad en su aplicación, conviene encomendar este servicio a una persona con elementos para coordinar los diversos aspectos.

Lo más importante en este aspecto de servicios es que el *staff* haga notar constantemente " Que no obra con autoridad propia, sino delegada", que lo hace " A nombre y representación de la línea", asumiendo la responsabilidad de conseguir en el mayor grado posible la autorización de los jefes de línea.

2.2 ORGANIGRAMAS

Estos representan en forma intuitiva y con objetividad los sistemas de organización. También suelen conocerse como cartas o gráficas de organización. Consisten en hojas en las que cada puesto de un jefe se representa por un cuadro que encierra el nombre de ese puesto y por medio de la unión de líneas se dan a conocer los canales de autoridad y responsabilidad.

Estos instrumentos son de mucha utilidad ya que nos revelan:

1) La división de funciones.

2) Los niveles jerárquicos.

3) Las líneas de autoridad y responsabilidad.

4) Los canales formales de la comunicación.

5) La naturaleza lineal o *staff* del departamento.

6) Los jefes de cada grupo de empleados, trabajadores.

7) Las relaciones que existen entre los diversos puestos de la empresa y en cada departamento o sección.

2.3 ELEMENTOS DE UNA ORGANIZACIÓN[11]

Las organizaciones en general deben de contar con ciertos elementos los cuales son considerados para su crecimiento y el desarrollo de sus productos, estos elementos garantizan el cumplimiento de los objetivos establecidos por la organización.

Los cimientos que soportan las organizaciones, los conforman los siguientes elementos :

a) Misión.

b) Metas.

c) Objetivos.

d) Políticas.

e) Norma.

2.3.1 Misión

Podríamos considerarla como la puesta en marcha de nuestras diferentes metas establecidas, apoyándonos en nuestros objetivos, para así tener una misión más clara y concreta. Por lo tanto es la razón de ser de toda organización.

2.3.2 Metas

Se puede considerar una meta como un resultado concreto y medible que se desea lograr. Acorde a los pasos previos en la metodología de calidad o como la declaración de una función especifica a corto plazo.

Hay tres niveles de metas:

1. El cumplimiento de requisitos (calidad explícita).
2. La satisfacción de expectativas (calidad implícita).
3. El proporcionar un nivel sorprendente al cliente.

Cabe señalar que si no se han fijado políticas, no se pueden establecer metas.

Determinada una política, la meta se hace vidente. Las metas deben formularse para cuestiones prioritarias únicamente, es decir, deben ser un número reducido.

Las metas deben expresarse de manera concreta y con cifras, para ser explicadas racionalmente; además se debe proporcionar a los empleados toda la información que necesiten. En nuestro caso, la información es proporcionada a los integrantes del equipo de diseño, evitando términos ambiguos que no producen buenas prácticas de control.

Las metas deben fijarse con base en resultados que la empresa desee alcanzar; es mejor formularlas asegurando la cooperación entre todas las divisiones, que asignando metas independientes para cada área. Por supuesto, hay que plantearlas siempre en base a datos y hechos.

Para la elaboración completa de las metas hay que buscar enunciados que determinen con precisión lo que debe realizar la compañía, el área o la persona dentro de un periodo concreto.

Algunas de las normas que debemos seguir para la elaboración de metas son las siguientes:

---

[11] En el ***Manual del participante. QFD Servicios,*** CENCADE, México, 1996.

- ➔ Definirlas en función de los resultados, condiciones por conquistar y no en función de los trabajos a ejecutar.

- ➔ Redactarlas de manera que puedan analizarse cada determinado tiempo.

- ➔ Manifestarlas en tiempos positivos, es decir, diciendo lo que se quiere lograr y no lo que se quiere evitar.

- ➔ Exponerlas en forma concisa y breve, no caer en descripciones minuciosas o complejas.

- ➔ Destinarlas a un solo resultado final y no obligarlas a contraer compromisos múltiples.

- ➔ Redactarlas en términos que puedan cuantificarse y sean fácilmente medibles.

Las metas son importantes, cuando menos, por cuatro motivos:

1. **LAS METAS PROPORCIONAN UN SENTIDO DE DIRECCIÓN.** Cuando no existe una meta, las personas y sus organizaciones suelen avanzar confundidas, reaccionando a los cambios de ambiente sin un sentido claro de lo que quieren lograr en realidad. Al establecer metas, las personas y sus organizaciones refuerzan sus motivos y obtienen una fuente de inspiración que les sirve para superar los obstáculos que se les presentan inevitablemente.

2. **LAS METAS PERMITEN ENFOCAR NUESTROS ESFUERZOS.** Todas las organizaciones, así como las personas, contamos con recursos limitados y una amplísima serie de posibilidades para usarlos. Al momento de elegir una meta, o una serie de metas relacionadas, establecemos prioridades y nos comprometemos con la forma que usaremos los recursos limitados.

3. **LAS METAS GUIAN NUESTROS PLANES Y DECISIONES.** Quizá nos debemos de cuestionar a nosotros mismos: ¿Cuál es nuestra meta? La respuesta a esta

pregunta dará forma a nuestros planes a corto y largo plazos, y nos servirá para tomar muchas decisiones fundamentales.

4. **LAS METAS SIRVEN PARA EVALUAR NUESTRO AVANCE.** Una meta definida con claridad, mensurable y con un límite de tiempo concreto se convierte en un parámetro de los resultados y permite a las personas y a los gerentes evaluar los avances logrados. Por lo tanto, las metas forman parte esencial del *control* que veremos en el capítulo IV; es decir, el proceso para asegurarse de que los actos se ajustan a nuestras metas y planes elaborados para alcanzarlas. Si descubrimos que nos estamos alejando de un curso o si surgen contingencias inesperadas, podremos tomar medidas correctivas modificando nuestro plan.

2.3.3 Objetivos

La palabra objetivo implica la idea de algo hacia lo cual tenemos dirigidas nuestras acciones. Puede también conocerse como meta.

Un objetivo representa lo que se espera alcanzar en el futuro como resultado del proceso administrativo. En el fondo, es material de la unidad de fin, esencial en todo un grupo social, ya que es aquello a lo que las acciones de todos se dirigen.

La importancia que tiene fijar objetivos es clara y precisa, pues ellos dan la razón de ser de una empresa y lograr resultados de máxima eficiencia.

2.3.4 Políticas

Las políticas son criterios generales que tienen por objeto orientar a la acción, dando a los jefes campo para la toma de decisiones que les corresponde tomar; sirven para formular, interpretar o suplir algunas normas concretas.

La importancia de las políticas estriba en que son indispensables para una adecuada delegación de responsabilidades.

Las políticas son el objetivo de acción; el objetivo fija metas, en tanto que las políticas imperan una orden para lanzarse a conseguirlas.

2.3.5 Norma

Es el procedimiento que se ajusta a nuestro desarrollo y modelo a que aspiramos, por lo cual deben de demostrar flexibilidad debido al comportamiento del individuo.

2.4 ESTRUCTURA DE GRUPOS PARA DESARROLLO

Dentro de la gran variedad de grupos de desarrollo software todos deben de contar con una estructura interna bien definida la cual puede estar dentro de las siguientes tres categorías:

1ª **Grupo democrático**

Dentro de este tipo de grupo todos y cada uno de sus integrantes forman parte de las discusiones generadas para poder tomar una decisión. Aquí, el liderazgo puede o no rotar; si el liderazgo rota, se le conocerá como un grupo sin egoísmo; pero si no rota, se le conoce como un grupo jerárquico democrático.

La estructura del grupo es la siguiente:[12]

GRUPO DEMOCRATICO

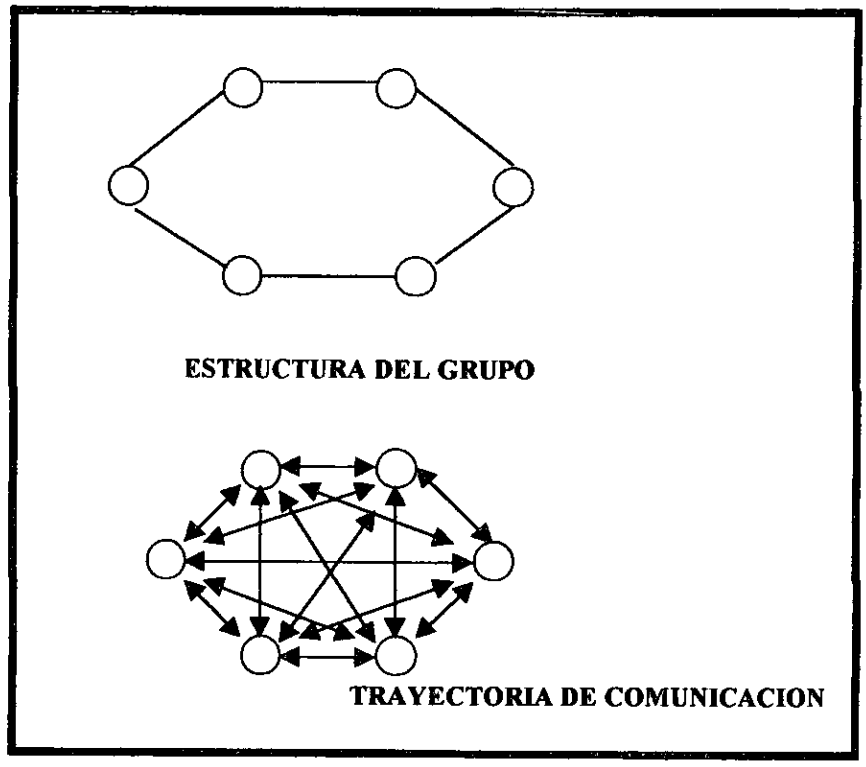

FIGURA 6

Ventajas:

√ Todos los elementos del grupo participan en la toma de decisiones.

√ La estructura del equipo permite que aprendan unos de otros.

√ Con esta estructura se logra un ambiente de trabajo confortable, permitiendo con esto una mayor integración en las funciones.

Desventajas :

- Existe el compromiso de que todos los miembros trabajen juntos.

[12] En los apuntes de la materia ***Calidad*** (décimo semestre).

- La falta de responsabilidad que pudiera ocurrir.
- La falta de iniciativa por parte de cada uno de los integrantes.

2ª Grupo con jefes de programación

En este tipo de grupo el jefe es el responsable de la planeación, desarrollo, servicios, y de todas las fases del desarrollo del sistema.

La estructura del equipo es la siguiente:[13]

GRUPO CON JEFES DE PROGRAMACIÓN

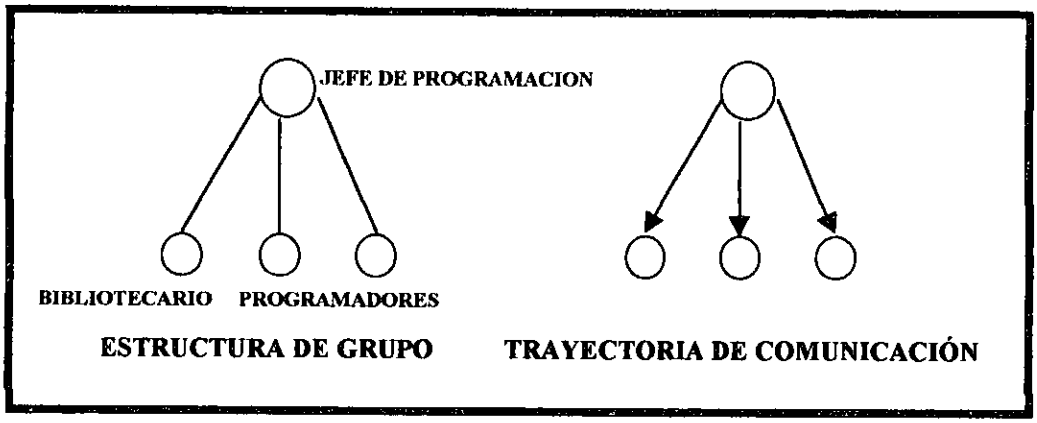

FIGURA 7

Ventajas:

√ Cada integrante tiene una función especifica por desarrollar.

√ El jefe revisa las tareas y reporta los avances.

√ El jefe es el encargado de tomar las decisiones.

[13] Idem.

Desventajas:

- Se reducen las trayectorias de la comunicación entre los elementos del equipo.
- El crecimiento y desarrollo del equipo depende de la eficiencia del jefe.

3ª Grupo bajo la jerarquía administrativa

Este grupo de trabajo es la combinación del grupo con jefes de programación y el grupo democrático, y tiene como límite el número de sus integrantes que va de dos a siete elementos.

La estructura del grupo es la siguiente:[14]

GRUPO BAJO LA JERARQUIA ADMINISTRATIVA

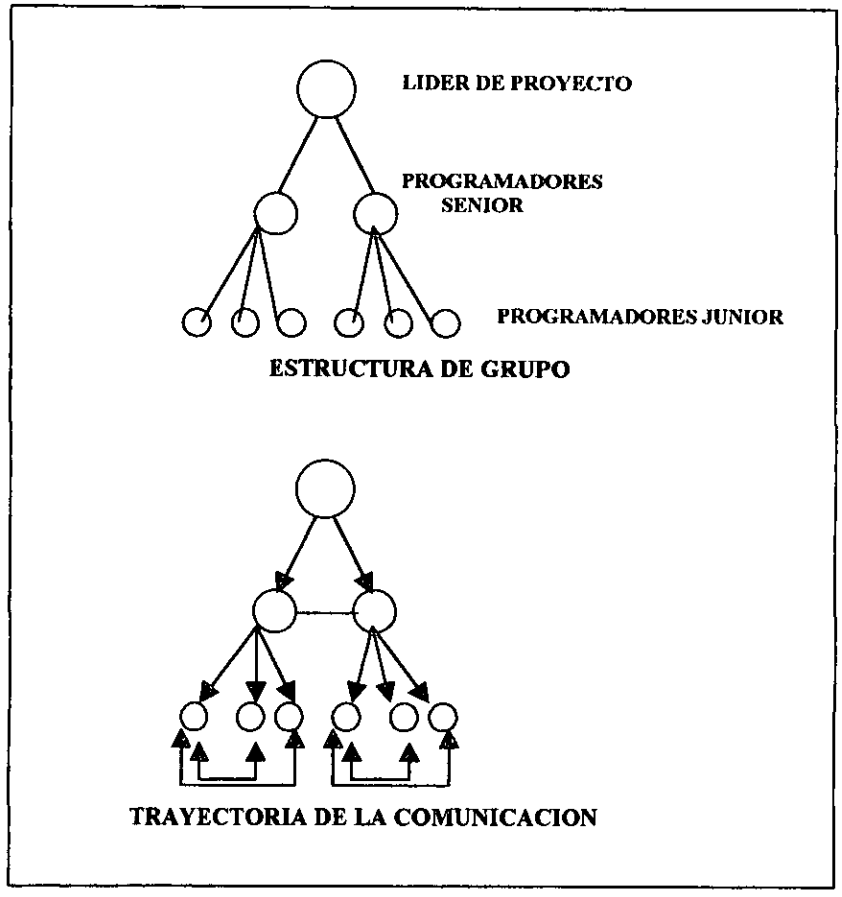

FIGURA 8

[14] Idem.

Ventajas:

√ El líder es el encargado de asignar las tareas.

√ Se pueden detectar mas fácilmente las área problema.

√ Se limita las trayectorias de la comunicación.

√ Existe un balance en la carga de trabajo (cada subsistema se le asigna a un equipo).

Desventajas:

- Los programadores técnicos más competentes tienden a ser promovidos hacia posiciones administrativas, lo cual proporciona prestigio y salario. La promoción anterior puede traer a un doble efecto negativo, ya que se puede perder un buen programador y ganar un mal administrador. Por lo tanto hay que tener cuidado con estos cambios pagarle a la gente por lo que hace y no por el puesto que ocupa.

CAPITULO III

PLANEACIÓN

Planeación- Importancia de la planeación- Planeación de una estructura organizacional- Modelos de planeación en el desarrollo del software

3.1 QUE ES LA PLANEACIÓN[15]

La planeación consiste, en fijar el curso concreto de acción que ha de seguirse, estableciendo los principios que habrán de orientarlo, la secuencia de operaciones para realizarlo y las determinaciones de tiempos y de números, necesarias para su realización.

Goetz dijo que planear es *"hacer que ocurran las cosas que, de otro modo, no habrían ocurrido"*. Esto equivale a trazar los planos para fijar dentro de ellos nuestra futura acción.

La planeación es una forma concreta de la toma de decisiones que aborda el futuro específico de una organización. Se puede decir que la planificación es la locomotora que arrastra el tren de actividades de la organización y el control.

La planeación no es sólo un hecho, con un principio y un final claros. Es un proceso continuo que refleja los cambios del ambiente entorno a cada organización y se adapta a ellos.

3.2 IMPORTANCIA DE LA PLANEACIÓN

En las organizaciones, la planeación es el proceso de establecer metas y elegir los medios para alcanzar dichas metas. Sin planes, los jefes de equipos de desarrollo no pueden saber cómo organizar a su personal ni a sus recursos debidamente; quizás incluso ni siquiera tengan una idea clara de *qué* deben organizar. Sin planes, no se puede dirigir con confianza ni esperar que los demás nos sigan; asimismo, no tenemos muchas posibilidades de alcanzar nuestras metas ni de saber cuándo y dónde se desvían.

Planear es tan importante como hacer, porque:

a) La eficiencia, obra de orden, no puede venir del acaso, de la improvisación;

[15] Ver en STONER, James, obra citada en la nota 4, y en PRESSMAN, Roger, obra citada en la nota 1.

b) Así como en la parte dinámica lo central es dirigir, en la mecánica el centro es planear.

c) El objetivo sería infecundo, si los planes no lo detallaran, para que pueda ser realizado íntegra y eficazmente.

d) Todo plan tiende a ser económico; desgraciadamente, no siempre lo parece, porque todo plan consume tiempo, que, por lo distante de su realización, puede parecer innecesario.

e) Todo control es imposible si no se compara con un plan previo. Sin planes, se trabaja a ciegas.

La planeación, por lo tanto, se considera parte fundamental para cualquier organización, trayendo consigo ventajas y algunas desventajas, presentadas a continuación:

Ventajas:

- Cuando planeamos, tenemos la posibilidad de calendarizar nuestras actividades, por lo tanto tenemos un mayor control de las mismas.
- Planeamos todas nuestras actividades a futuro.
- Dentro de nuestra planeación existe una fecha de inicio y una de termino, lo cual nos ayuda a tener fechas de entrega a tiempo.
- El personal, parte fundamental de toda organización, se motiva en la participación del proyecto al tener planeación.

Desventajas :

- Existe el riesgo que el personal no tenga la responsabilidad necesaria, arrastrando así la falta de compromiso e iniciativa personal.
- Al tener nuestras fechas planeadas, los integrantes pierden su libertad y se sienten comprometidos.
- Otras de las desventajas son las políticas y reglas con las que cuenta la empresa, que se ven afectadas con la planeación.

3.3 PLANEACIÓN DE UNA ESTRUCTURA ORGANIZACIONAL

Las planeación es uno de los elementos mas importantes dentro de la estructura organizacional de las empresas.

La utilización de la planeación dentro del desarrollo de software es indispensable ya que con esta alcanzaremos a visualizar de manera mas clara las fechas establecidas para el cumplimiento de nuestros objetivos, así como el avance de nuestras actividades y el presupuesto.

Dentro del ciclo de vida de un producto es indispensable realizar actividades, dentro de las cuales están las siguientes:[16]

1) **Planeación:** es la encargada de identificar a los clientes externos, así como sus necesidades y factibilidad, además de ofrecer supervisión del desarrollo del producto de principio a fin.

2) **Desarrollo:** este es el encargado de especificar, diseñar e instrumentar; después de estos pasos realiza la depuración, las pruebas y la integración del producto.

3) **Servicios:** aquí se proporcionan las herramientas automatizadas y recursos computacionales para todas las actividades, y efectúa la administración y distribución del producto, además de ofrecer distintos apoyos administrativos.

4) **Publicación:** se encarga de la elaboración de manuales de usuario, instrucciones de instalación y los principios de operación.

5) **Control total de la calidad:** encargado de la evaluación del código fuente y de las publicaciones antes de su entrega a los clientes.

6) **Mantenimiento:** es el encargado de llevar a cabo la corrección de errores y a la vez hacer mejoras durante la vida de un producto y adaptarlo a nuevos ambientes de procesamiento.

---

[16] En los apuntes de la materia ***Organización de centros de cómputo*** (octavo semestre)-

Toda la planeación de las estructuras organizacionales se refieren a la forma en que se dividen, agrupan y coordinan las actividades de la organización en cuanto a los desarrolladores de sistemas, que bien pueden ser el analista de sistemas, programadores o jefes de equipo.

Existen algunos métodos para poder llevar a cabo estas tareas, dentro de las cuales se pueden destacar las siguientes:

* **Formato de proyecto**

En este formato se integra un grupo de programadores y analistas, encargados de llevar el proyecto de principio a fin y que realizan la definición, instrumentación, prueba y la revisión del producto, así como el desarrollo de los documentos de apoyo. La estructura del formato se representa en la figura 9.

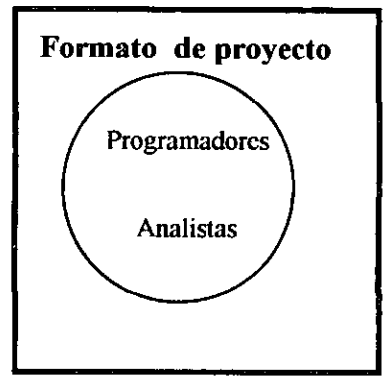

Figura 9

* **Formato funcional**

Aquí se integran equipos que se especializan en diversas tareas del ciclo de la vida del producto; esto se hace conforme se avanza el desarrollo del mismo y así va pasando de un equipo a otro conforme el producto evoluciona.

Este formato presenta las siguientes características:

- Permite la especialización en ciertas áreas del ciclo de vida del producto.
- Es indispensable una comunicación muy estrecha entre los diferentes equipos.
- Es conveniente que los miembros de los equipos se roten entre sí, con el fin de obtener el conocimiento y valorar la importancia de otras áreas.

El formato funcional es utiliza con eficiencia los recursos de la especialización de cada uno de sus grupos. Pudiendo tomar como ventaja que la supervisión se hace muy fácil, ya que cada uno de los grupos debe de ser experto en una gama limitada de actividades, además de facilitar el movimiento de las actividades especializadas, para ser usadas en los puntos donde más se necesitan. La estructura del formato se representa en la figura 10.

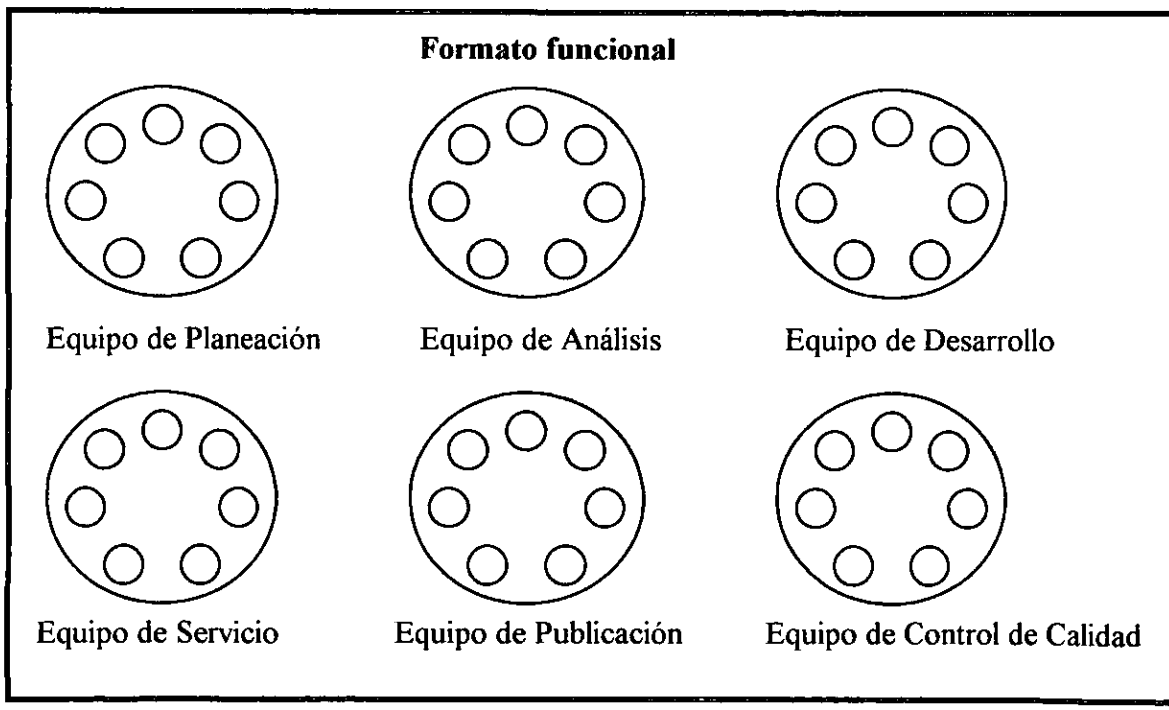

Figura 10

* **Formato matricial**

En este tipo de formato cada función tiene su propia administración y un equipo exclusivamente dedicado a esa función. (ver figura 11).

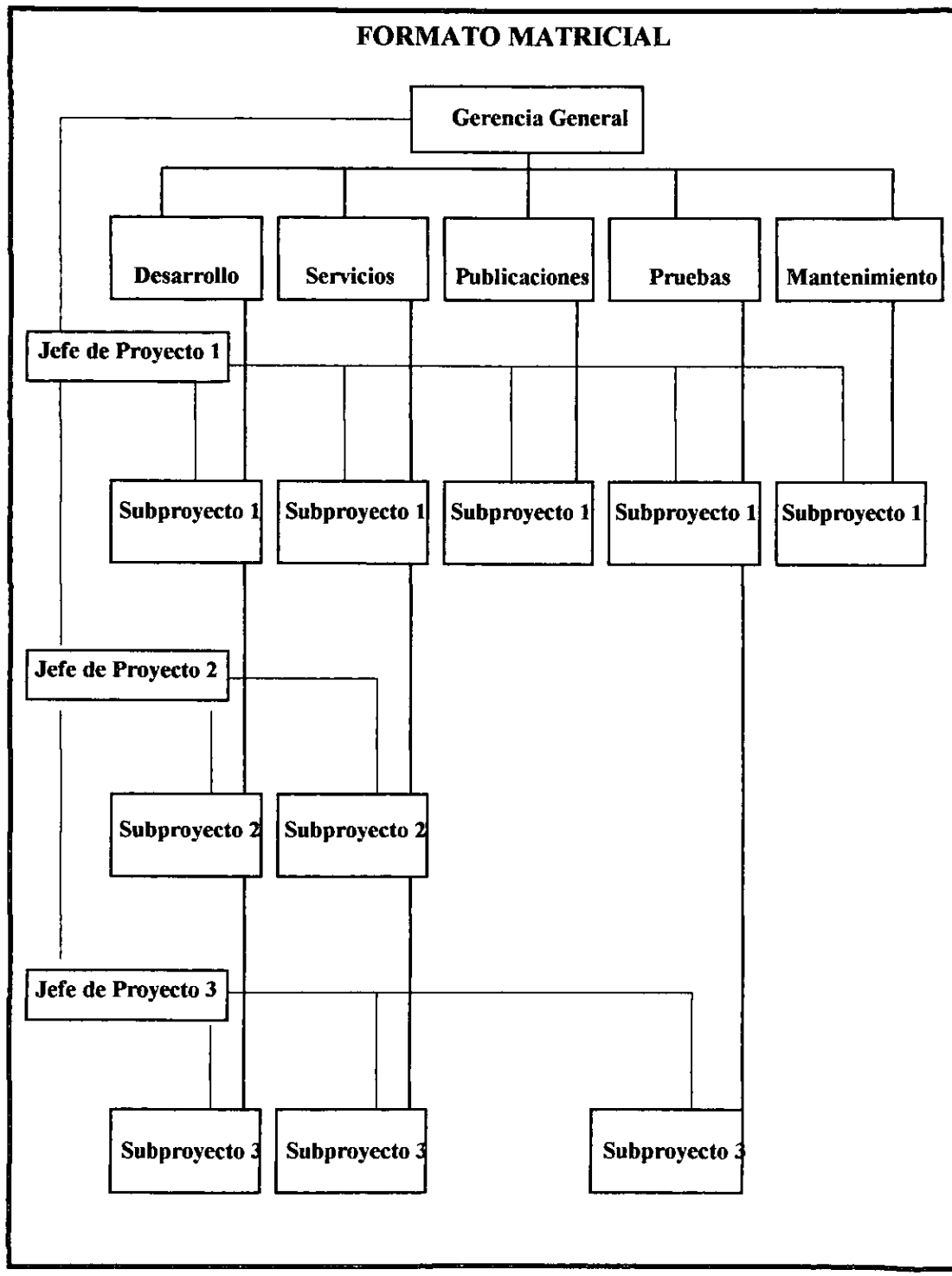

Figura 11

Aquí cada proyecto tiene un administrador, que organizacionalmente es miembro de la planeación, desarrollo, diseño, pruebas de producto, instrumentación.

Este formato tiene las siguientes características:

a) Los participantes tienen dos jefes, uno encargado de la función y otro encargado del proyecto.

b) Los participantes están dedicados a una sola función la cual los hace especialistas en las mismas, permitiendo que hagan su trabajo con mayor eficiencia.

c) Los participantes pueden pasar de un proyecto a otro sin problemas, siempre y cuando hayan terminado el anterior.

3.4 MODELOS DE PLANEACIÓN EN EL DESARROLLO DE SOFTWARE

Hoy en día, para poder resolver los problemas reales de la industria del desarrollo de software, el equipo de desarrollo o ingeniero de software deben incorporar estrategias de desarrollo que acompañen al proceso, métodos, herramientas y etapas.

En esta sección se tratarán diferentes modelos de procesos para el desarrollo de software. Cada uno de los modelos presenta un intento de ordenar las actividades de diseño, recordando que cada uno de los modelos se caracterizará, junto con el control y la coordinación, de un proyecto de software real.

3.4.1 El modelo lineal secuencial

Llamado también modelo en cascada o ciclo de vida básico por la ingeniería de software, este modelo requiere de un enfoque que sea sistemático, secuencial del desarrollo del software, comenzando en el nivel de sistemas y progresando con el análisis, diseño, codificación, pruebas y mantenimiento; cada uno se describe a continuación:

- ***Ingeniería y modelado de sistemas e información***

El software siempre formará parte de un sistema mayor. El primer trabajo que se tiene es el establecimiento de requisitos que incluyan todos los elementos del sistema, asignando así al software algún subgrupo de dichos requisitos. Hay que tener esta visión cuando el software se deba interconectar con otros elementos como las bases de datos, personas o hardware. Esta es acompañada a su vez con una pequeña parte del análisis y diseño. La ingeniería de información acompaña a los requisitos recogidos en el nivel estratégico de la empresa y el área de negocios.

- ***Análisis de los requisitos del software***

Este proceso, como su nombre lo indica, es el análisis intensificado del software; esto es, la comprensión de la naturaleza de los programas a construirse. Aquí, el ingeniero o analista del software se ve comprometido a comprender el dominio de información, así como la función que se requiere, el comportamiento, rendimiento e interconexión.

- ***Diseño***

Este proceso consta de muchos pasos, pero se centra en cuatro atributos de un programa, como lo son: la estructura de datos, arquitectura del software, representación de interfaz y el

detalle procedimental (algoritmo). Este proceso traduce los requisitos en una representación del software que se pueda evaluar por calidad antes de comenzar la generación del código.

- ***Generación de código***

La generación de código es traducir el diseño del software en un lenguaje que la computadora pueda interpretar, y de igual manera lo entienda el ingeniero de software.

- ***Pruebas***

El proceso de pruebas se centra en los procesos lógicos internos del software, es decir, la realización de las pruebas para la detección de errores y el sentirse seguro de que la entrada definida produzca resultados reales de acuerdo con los resultados requeridos.

- ***Mantenimiento***

Inevitablemente, el software sufre cambios al ser entregado al cliente. Se producirán cambios al encontrar algunos errores, o porque el software debe adaptarse para acoplarse a los cambios de su entorno externo. Con esto me refiero a un cambio debido a un sistema operativo o dispositivo periférico nuevo, o porque el cliente requiere algunas mejoras de rendimiento o funcionales.

Este modelo es el paradigma más antiguo y utilizado en el diseño de software; pero tanto se ha criticado este paradigma, que pone en duda la eficacia del mismo.

Ahora bien, se han encontrado algunos problemas al utilizar el modelo lineal secuencial. los errores que se mencionaran, son errores reales, pero aun así el ciclo de vida clásico sigue siendo el modelo de proceso más usado por la ingeniería del software. Pese a tener

debilidades, es significativamente mejor que un enfoque hecho al azar para el desarrollo del software.

El modelo lineal secuencial, por ser el primero utilizado en la ingeniería del software, muestra algunas desventajas al compararlo con los nuevos modelos, en las que se incluyen las siguientes:

1. Aunque es un modelo que se acopla a la interacción este lo hace indirectamente. Como resultado, los cambios pueden causar confusión cuando empieza a trabajar el equipo de proyecto.

2. Es necesario para este modelo que nuestro cliente exponga de manera explícita todos los requisitos, de otro modo tendremos dificultades al comienzo de nuestros proyectos.

3. En el modelo el cliente debe de tener paciencia, ya que una versión de trabajo del (los) programa (s) se le podrá presentar hasta que el proyecto este muy avanzado.

4. Al no detectar los errores desde el principio puede ser desastroso.

5. En este modelo se tiene muy marcado los "estados de bloqueo" en el que algunos miembros del equipo de proyecto deben esperar a otros miembros del equipo para completar tareas pendientes, causando un sobrepaso en el tiempo que se emplea en el trabajo productivo.

Estos estados de bloqueo son muy marcados al principio y al final del proceso del modelo lineal secuencial

(VER FIGURA 12.)

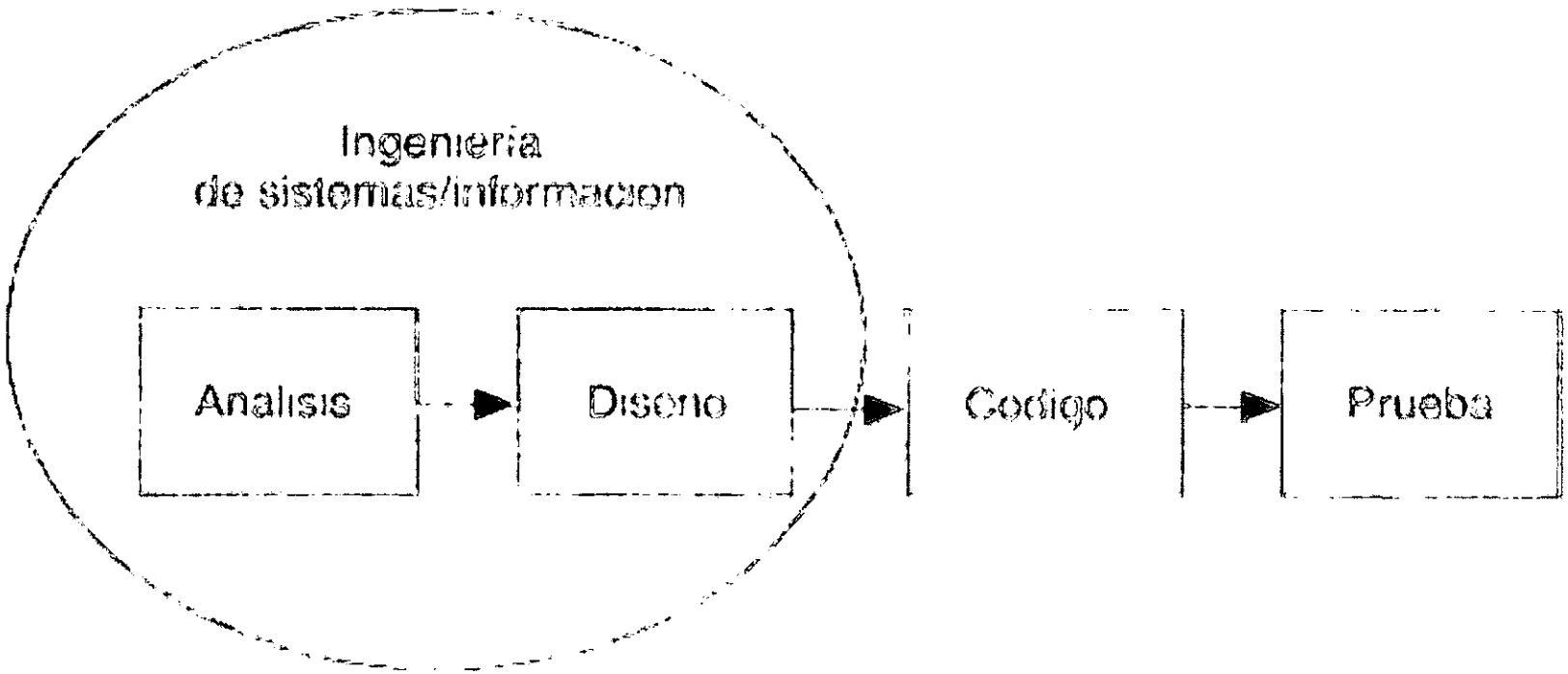
Ingeniería
de sistemas/informacion
Analisis
Diseno
Codigo
Prueba

Figura 12

3.4.2 El modelo de construcción de prototipos

Frecuentemente nos encontramos con dos grandes problemas: el primero es que el cliente no pueda identificar los requerimientos detallados de entrada, procesamiento, o de salida, o el otro caso es cuando el responsable del desarrollo del software pueda o no estar seguro de la eficacia de su algoritmo, de la capacidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción del hombre con la máquina. Para este tipo de situaciones el modelo de construcción de prototipos permite un mejor enfoque.

Este modelo comienza sus paradigmas con la recolección de requisitos, identificados por el desarrollador y el cliente, así como los objetivos globales y las áreas del esquema; es así donde nace un diseño rápido. El diseño rápido está centrado en la representación de esos aspectos que serán visibles para el usuario; ya sean los enfoques de entrada, los formatos de salida, el diseño rápido lleva a la construcción de un prototipo; el prototipo es evaluado por el cliente y el usuario y lo utilizan para refinar los requisitos del software a desarrollar.

Lo ideal de este modelo es que el prototipo funcione como un identificador de los requisitos del software.

El prototipo puede servir como primer sistema. Es verdad que a los desarrolladores y a los clientes les gusta el modelo de construcción de prototipos. A los usuarios les gusta el sistema real y a los que desarrollan les gusta construir algo inmediatamente. Sin embargo, al igual que todos los modelos, éste puede causar problemas; esto, debido a las siguientes razones:

1. El cliente ve lo que parece ser una versión de trabajo del software sin saber que se ha elaborado con prisa, dejando atrás la calidad del software global la facilidad de mantenimiento a largo plazo cuando se le informa al cliente que el producto debe construirse otra vez para poder mantener los niveles de calidad muy en alto, el cliente no lo entiende y pide al desarrollador que sólo se le apliquen unos pequeños ajustes y así hacer del prototipo un diseño final.

2. El desarrollador hace compromisos de implementación para hacer que el prototipo funcione rápidamente. Con toda esa presión de tiempo, el desarrollador puede no darse cuenta de no utilizar un sistema operativo o lenguaje de programación adecuado o usar con los que cuenta el cliente simplemente porque están disponibles. Con la experiencia, el desarrollador se familiariza con estas selecciones y olvida las razones por las que fue inadecuada.

La clave es definir las reglas del juego al iniciar el diseño. Dicho de otra forma: que tanto el cliente como el desarrollador se deben de poner de acuerdo en que el prototipo se construya para servir como un mecanismo de definición de requisitos; es así como se realiza la ingeniería del software, con una visión a la calidad y facilidad de mantenimiento. (VER FIGURA 13.)

Este modelo no tiene su excepción en cuanto a ventajas y desventajas por lo cual presenta las siguientes.

Ventajas :

- Cuando obtenemos un diseño rápido, en ingeniero de sistemas se centra en una representación de aspectos del software que sean visibles para el usuario y el cliente.
- Al desarrollar un prototipo, el desarrollador intenta hacer uso de los fragmentos del programa ya existentes o de herramientas, permitiendo generar así programas de trabajo.

Desventajas :

1. El cliente ve lo que parece ser una versión de trabajo de software (prototipo) sin darse cuenta que por la prisa de hacerlo funcionar no se ha tomado en cuenta la calidad y al informarle que se debe de elaborar otra vez para poder obtener los niveles de calidad adecuados, el cliente no lo entiende y pide solo unos ajustes para así poder hacer de un prototipo un producto final.
2. El compromiso de implementación para hacer que el prototipo funcione rápidamente lleva al desarrollador a utilizar un sistema operativo o un lenguaje de programación inadecuado, solo por que lo conoce o es lo que tiene disponible.

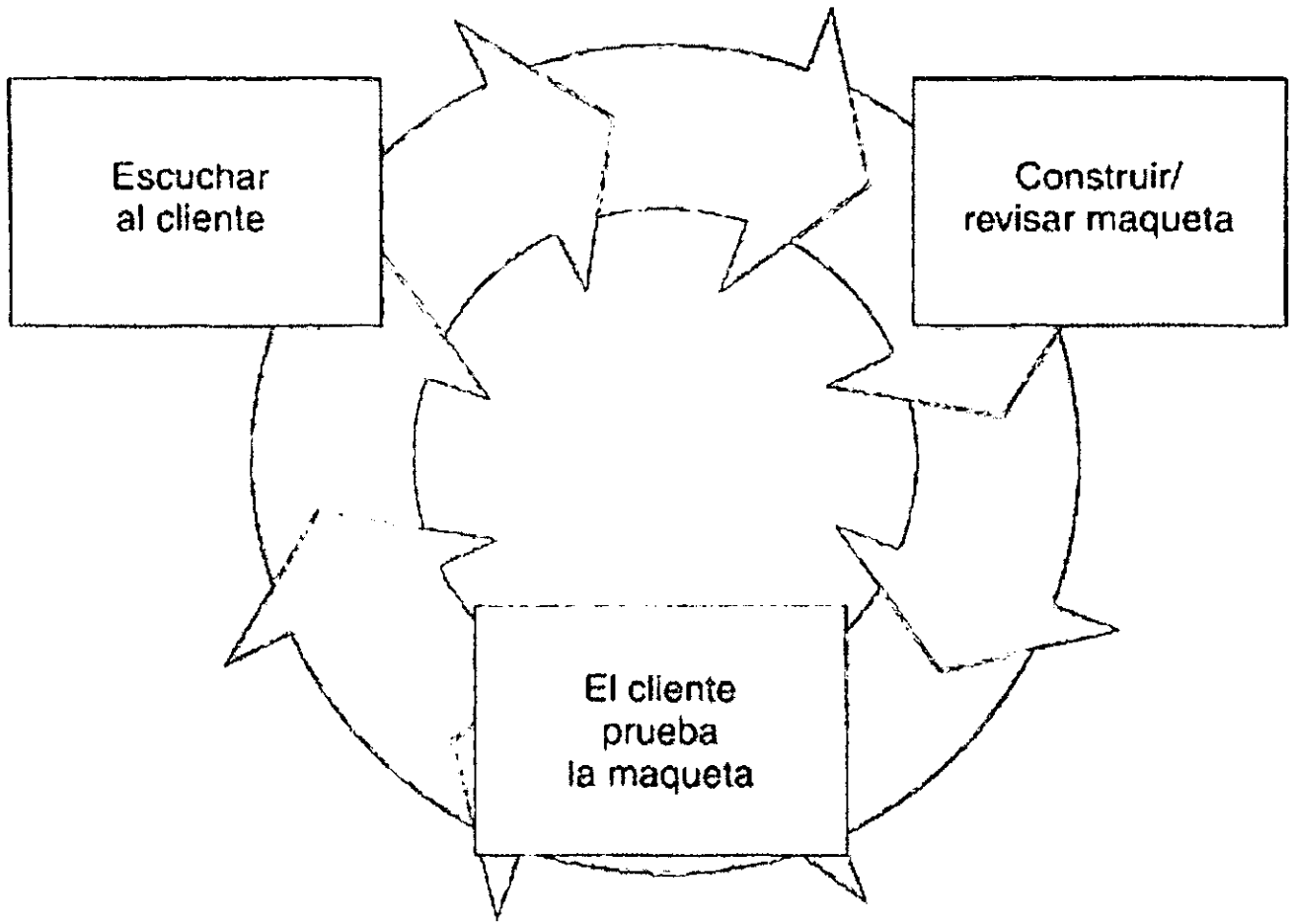
Escuchar
al cliente
Construir/
revisar maqueta
El cliente
prueba
la maqueta

Figura 13

3.4.3 El modelo de DRA

El Desarrollo Rápido de Aplicaciones (DRA) es un modelo de proceso del desarrollo del software lineal secuencial, su ciclo de vida es extremadamente corto y es una adaptación de alta velocidad del modelo lineal secuencial. El modelo de DRA permite desarrollar un sistema completamente funcional durante periodos muy cortos.

El enfoque del DRA comprende las siguientes faces:

- ***Modelado de gestión***

Se refiere al tipo de flujo de información entre las funciones de gestión; estas se modelan de acuerdo a preguntas básicas como: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la genera?

- ***Modelado de datos***

Este se define como las características llamadas atributos de cada uno de los objetos y las relaciones entre esos objetos.

- ***Modelado de proceso***

Aquí los objetos de datos se transforman para lograr un flujo de información que es indispensable para implementar una función de gestión. Las descripciones de todo el proceso son creadas para poder añadir, modificar o suprimir los objetos de datos.

- ***Generación de aplicaciones***

El DRA asume la utilización de técnicas de cuarta generación; trabaja para volver a utilizar componentes de programas ya existentes a crear componentes reutilizables. En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

➢ ***Pruebas y entregas***

El DRA simpatiza mucho con la reutilización de componentes de programas, ya se han comprobado muchos de los componentes de los programas; esto reduce el tiempo de pruebas.

El proceso del DRA demanda ámbitos en escalas. Si la aplicación de gestión puede modularse de forma que permita completarse a cada una de las funciones principales, en menos de tres meses es un buen candidato para el DRA. Cada una de las funciones pueden ser afrontadas por un equipo de DRA diferente y ser integradas en un solo conjunto.

De igual forma, el DRA presenta inconvenientes para el desarrollo de software, las cuales se enuncian a continuación:

- Al enfocarnos a proyectos muy grandes, aun siendo por escalas, este modelo necesita recursos humanos suficientes como para crear el número suficiente de equipos.
- El modelo requiere clientes y desarrolladores que se comprometan en las rápidas actividades indispensables para poder complementar un sistema en un marco de tiempo abreviado; al no haber el compromiso necesario de por medio, el modelo fracasará.

El modelo de DRA presenta las siguientes ventajas y desventajas.

Ventajas :

1. Con este modelo se puede lograr el desarrollo rápido utilizando un enfoque de construcción basado en componentes.
2. El DRA permite crear un "sistema completamente funcional".
3. Reutiliza componentes que habían sido utilizados en programas ya existentes.
4. Reduce el tiempo de pruebas, ya que muchos de los componentes de los programas que son reutilizados ya se habían revisado.

Desventajas :

1. Requiere de recursos humanos suficientes para poder crear el número completo de equipos de DRA.
2. Requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado
3. El modelo de DRA no es adecuado cuando los riesgos técnicos son altos.

Podemos concluir que si no podemos sincronizar adecuadamente la construcción de los componentes necesarios para el DRA, éste será muy problemático; por lo tanto, no es adecuado cuando los riesgos técnicos son muy altos. (VER FIGURA 14.)

3.4.4 El modelo incremental

Este modelo consiste básicamente en una combinación de los elementos del modelo lineal secuencial, con la filosofía interactiva de la construcción de prototipos. Como lo muestra la FIGURA 15, en el modelo incremental se aplican secuencias lineales, al mismo tiempo que progresa el tiempo en el calendario y cada secuencia lineal produce satisfactoriamente un incremento en el software.

Al momento de utilizar este modelo, el primer incremento es esencial, por lo cual lo denominamos como núcleo; es así cuando llega la hora de evaluar el primer incremento, sufriendo una revisión detallada y se desarrolla un plan para el incremento siguiente.

Este modelo se considera interactivo, al igual que el modelo de construcción de prototipos; pero con la diferencia de que el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos se consideran versiones desmontadas del producto final, pero éstas nos proporcionan la capacidad que sirve al usuario y como una pequeña evaluación por parte del usuario.

El modelo DRA.

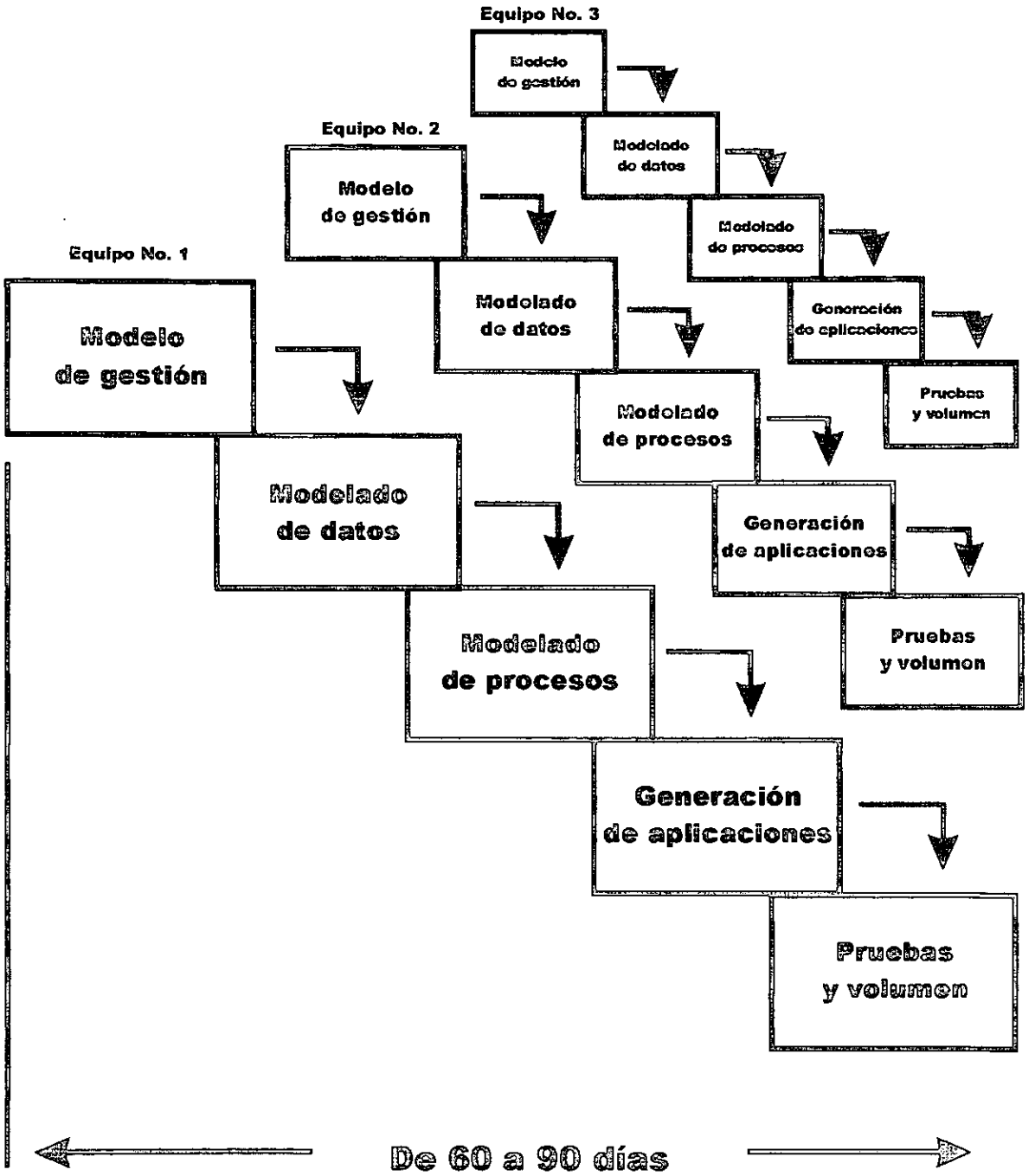

Figura 14

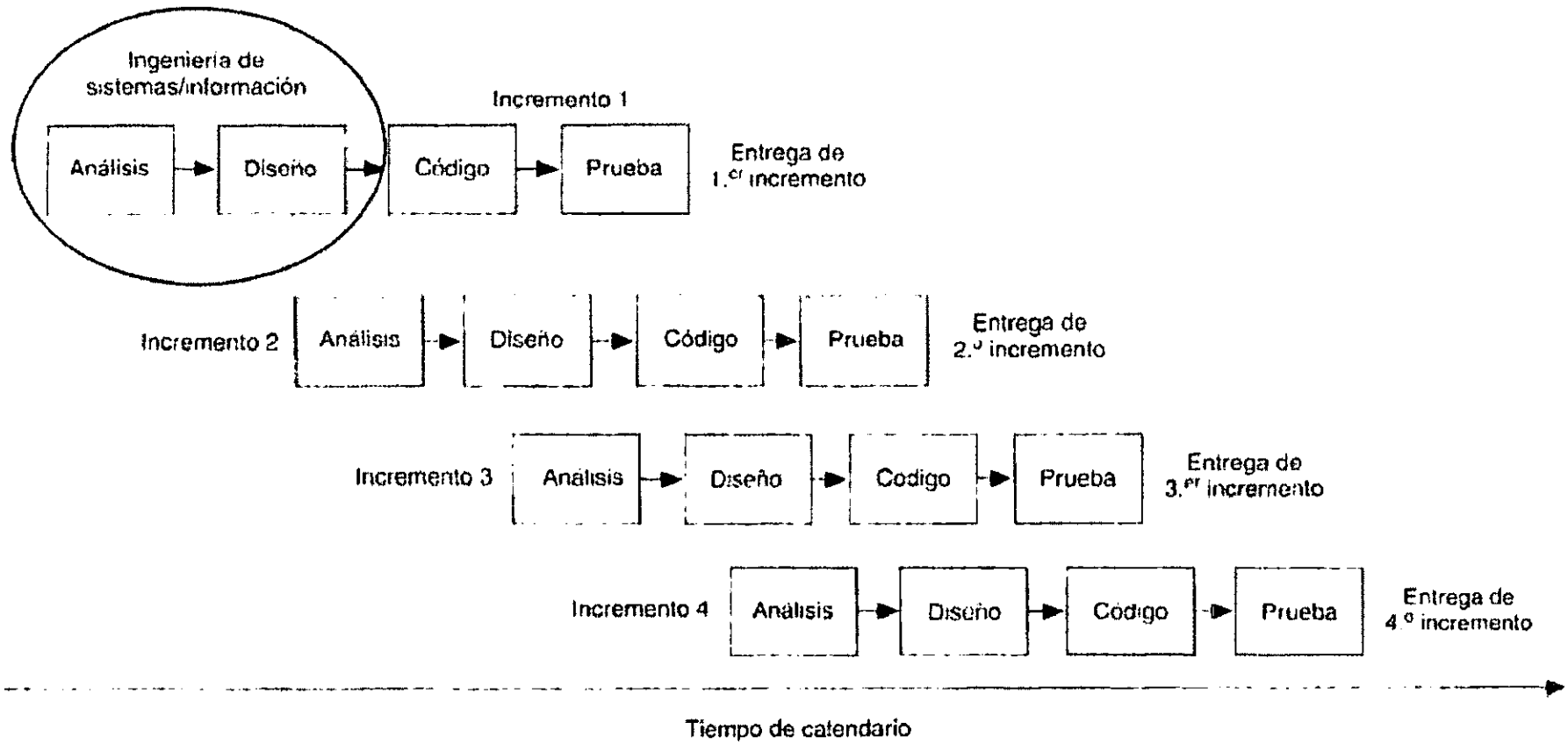
Ingeniería de sistemas/información
Incremento 1
Análisis
Diseño
Código
Prueba
Entrega de 1.er incremento
Incremento 2
Análisis
Diseño
Código
Prueba
Entrega de 2.º incremento
Incremento 3
Análisis
Diseño
Codigo
Prueba
Entrega de 3.er incremento
Incremento 4
Análisis
Diseño
Código
Prueba
Entrega de 4.º incremento
Tiempo de calendario

Figura 15

Este modelo de desarrollo incremental es recomendable cuando la cantidad de personal no es abundante, ya que los primeros incrementos emplean menos personas; pero si el núcleo es bien recibido, podemos añadir más personal para realizar el incremento siguiente.

Ventajas :

1. El modelo incremental se centra en la entrega de un producto operacional, garantizando así la satisfacción del cliente en cada incremento.
2. Existe un orden en el desarrollo de actividades dentro de cada incremento.

Desventajas :

1. No existe tiempo limite de terminación.
2. Si nuestro cliente no esta satisfecho con el primer incremento, no podemos pasar al siguiente incremento. Lo cual permite el cumplimiento del desarrollo de acuerdo a las especificaciones del cliente.

3.4.5 El modelo en espiral

El modelo en espiral es un modelo de proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. En este modelo, el software se desarrolla en versiones incrementales del software. Dentro de las primeras iteraciones, el modelo se podría trabajar en papel o como un prototipo, incrementando así hasta que las últimas versiones sean cada vez más completas para la ingeniería del software.

Este modelo se divide en tareas, las cuales podemos llamar regiones de tareas; esto, por la manera en que se divide el espiral. Característicamente existen entre tres y seis regiones, como lo muestra la FIGURA 16.

Dichas regiones son las siguientes:

- ***Comunicación con el cliente***

En esta fase se establece la comunicación con el cliente y se determinan los requerimientos del producto que demanda.

- ***Planificación***

Esta tarea es la requerida para definir los recursos con los que cuenta el cliente y el desarrollador, el tiempo de desarrollo, las fechas de inicio y entrega, y otras informaciones que estén relacionadas con el proyecto.

- ***Análisis de riesgos***

Aquí es posible evaluar los posibles riesgos técnicos y de gestión del desarrollo del sistema esto dependerá de la capacidad que tenga el ingeniero de software para poder visualizar a futuro los posibles riesgos.

- ***Ingeniería***

Esta etapa del espiral es la requerida para construir las representaciones de la aplicación.

- ***Construcción y adaptación***

En esta etapa se define las tareas para construir, probar, instalar y proporcionar soporte al usuario; un ejemplo citable podría ser la documentación y la práctica.

- ***Evaluación del cliente***

Con esta etapa lo que buscamos es la reacción del cliente, según la evaluación de las presentaciones del software creadas durante la etapa de ingeniería y la implementación de la instalación.

Cada región del espiral se adapta a las características del proyecto que se emprenderá, observando así que para proyectos pequeños el número de etapas y formalidad es bajo; sin embargo, para proyectos mayores y críticos, cada una de las regiones contiene tareas definidas con el fin de lograr un nivel más alto de formalidad.

Al empezar el proceso, el equipo de desarrollo girará en sentido de las manecillas del reloj, comenzando por el centro. En el primer circuito se produce el desarrollo de la especificación de productos, y los siguientes pasos se podrían usar para construir prototipos y así ir creando versiones más sofisticadas del software. Cada etapa de la planificación del proceso produce ajustes al proyecto.

El modelo clásico del espiral, termina al hacer la entrega del sistema; pero éste se puede adaptar y aplicar a lo largo de la vida del sistema, como se muestra en la FIGURA 17, donde definimos el punto de entrada del proyecto. Cada cubo situado a lo largo del eje representa el punto de arranque de cada proyecto.

Un proyecto de desarrollo de conceptos comienza en el centro del espiral y continúa hasta que completa el desarrollo del concepto. Si el desarrollo de concepto se desarrolla dentro de un producto real, el proceso procede al siguiente cubo, iniciando así un nuevo proyecto de desarrollo; este producto nuevo irá evolucionando a través del espiral, siguiendo el camino que se limita más brillante que en el centro.

Un modelo en espiral típico

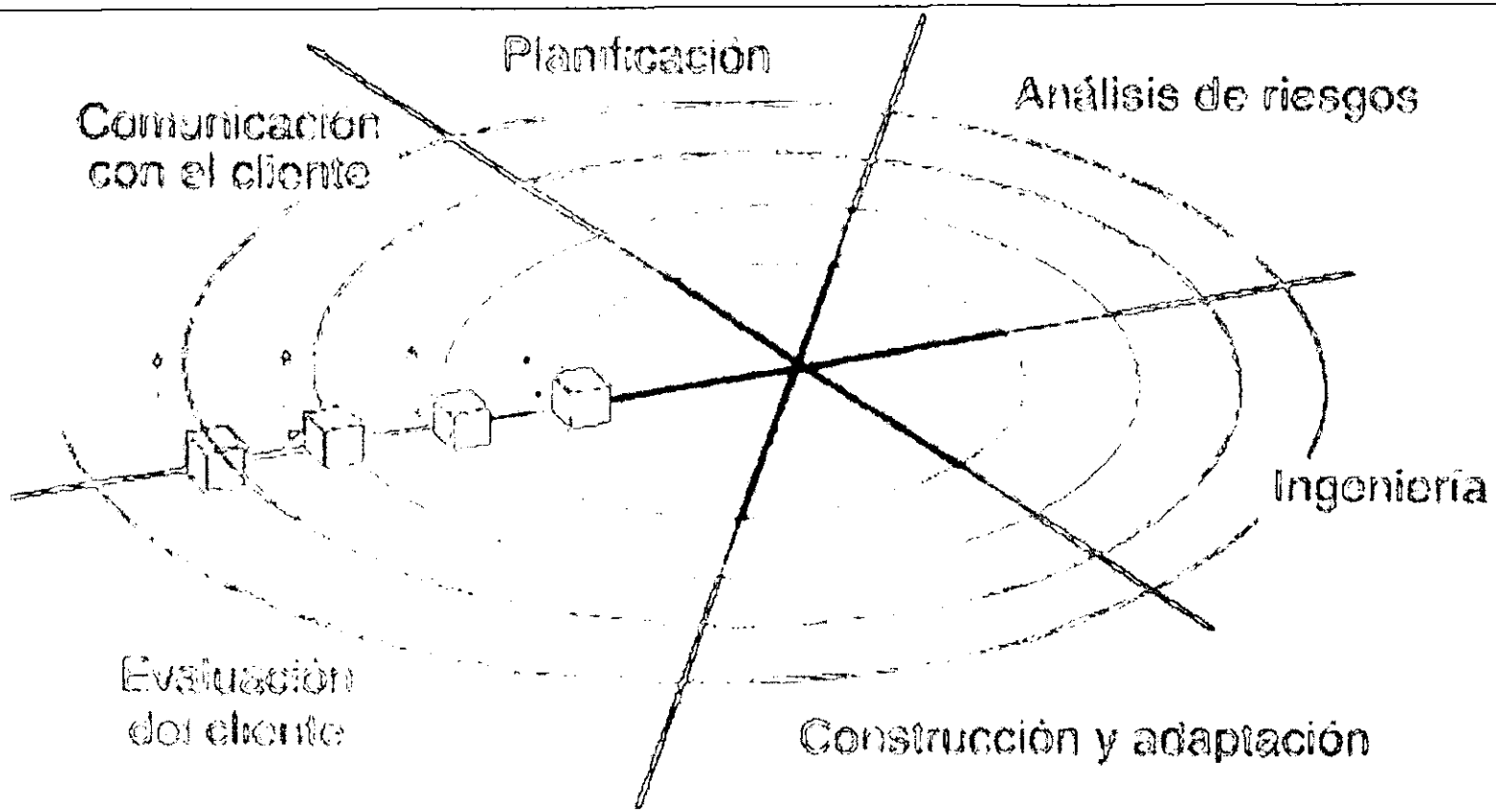

Figura 16

Modelo en espiral adaptado para el ciclo de vida clásico completo

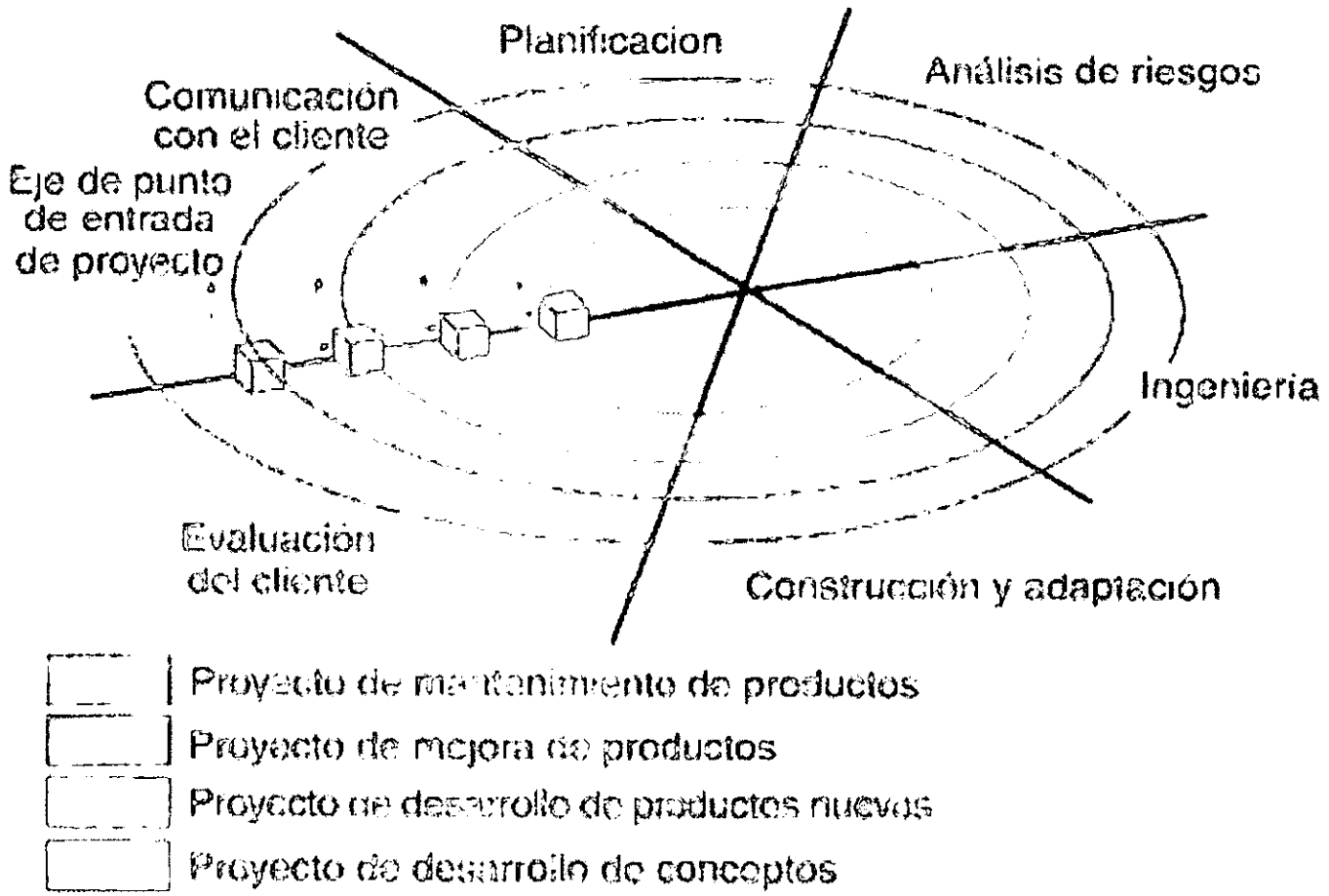

Figura 17

Este espiral adecuado al ciclo de vida del sistema permanece en forma operativa hasta que el software se retira.

Podríamos decir que este modelo es de los más realistas, ya que evoluciona a la par del proceso ocasionando que el desarrollador y el cliente comprendan y reaccionen mejor ante los riesgos que tiene cada nivel evolutivo. Este modelo, al estar basado en la construcción de prototipos, permite al desarrollador aplicar un enfoque de construcción de prototipos en cualquier etapa de evolución del producto.

Podría resultar difícil convencer a grandes clientes de que el enfoque evolutivo es controlable, ya que requiere una extraordinaria habilidad para la evolución del riesgo; pero teniendo esta habilidad podríamos llegar al éxito. Este modelo es relativamente nuevo, por lo que no se ha utilizado tanto como el modelo lineal secuencial o como el de construcción de prototipos.

Existen ventajas al utilizar el modelo en espiral, al igual que algunas desventajas las cuales se mencionan a continuación.

Ventajas :

1. El cliente al igual que el desarrollador comprenden y reaccionan mejor ante los riesgos en cada uno de los niveles evolutivos.
2. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos.
3. Demanda consideración de riesgos técnicos en cada una de las etapas del proceso.

Desventajas :

1. En este modelo es difícil el poder convencer a grandes clientes de que el enfoque evolutivo es confiable.
2. Se requiere una habilidad considerable para poder evaluar riesgos, con la que el modelo si cuenta.

3.4.6 El modelo de ensamblaje de componentes

Este modelo incorpora características del modelo de espiral, ya que también es evolutivo, exigiendo así un enfoque interactivo para el desarrollo del software.

El modelo de componentes configura las aplicaciones de componentes preparados de software llamados procedimientos.

La actividad de ingeniería comienza con la identificación de procedimientos candidatos, llevando a cabo esta actividad con la examinación de los datos que se van a manejar en la aplicación.

Los procedimientos que han sido creadas con anterioridad para otros proyectos de ingeniería se almacenan en una biblioteca de clases definida comúnmente como depósito. Una vez que se identifican los procedimientos candidatos, recurrimos a la biblioteca de procedimientos para examinar y determinar si los procedimientos ya existen. En caso de ser así, se extraen de la biblioteca y se adecuan para nuestro nuevo sistema. Si un procedimiento no lo tenemos disponible en la biblioteca, tendrá que elaborarse. Mediante los procedimientos extraídos de la biblioteca y los nuevos construidos para cumplir las necesidades únicas de la aplicación, componemos la primera parte de la interacción de aplicación a construirse.

Nuestros resultados al utilizar este modelo se basan en la gama de procedimientos que tengamos en la biblioteca; pero no hay duda que el ensamblaje de componentes nos proporciona ventajas significativas para los ingenieros del software. (VER FIGURA 18.)

Dentro de las ventajas que proporciona el modelo se enumeran las siguientes :

1. Reduce el tiempo que empleamos en un desarrollo hasta un 70%.
2. Existe una reducción muy alta de costos del proyecto pudiendo llegar hasta un 84%.
3. Tiene una productividad del 26.2 % contra el establecido por la industria que es del 16.9%.

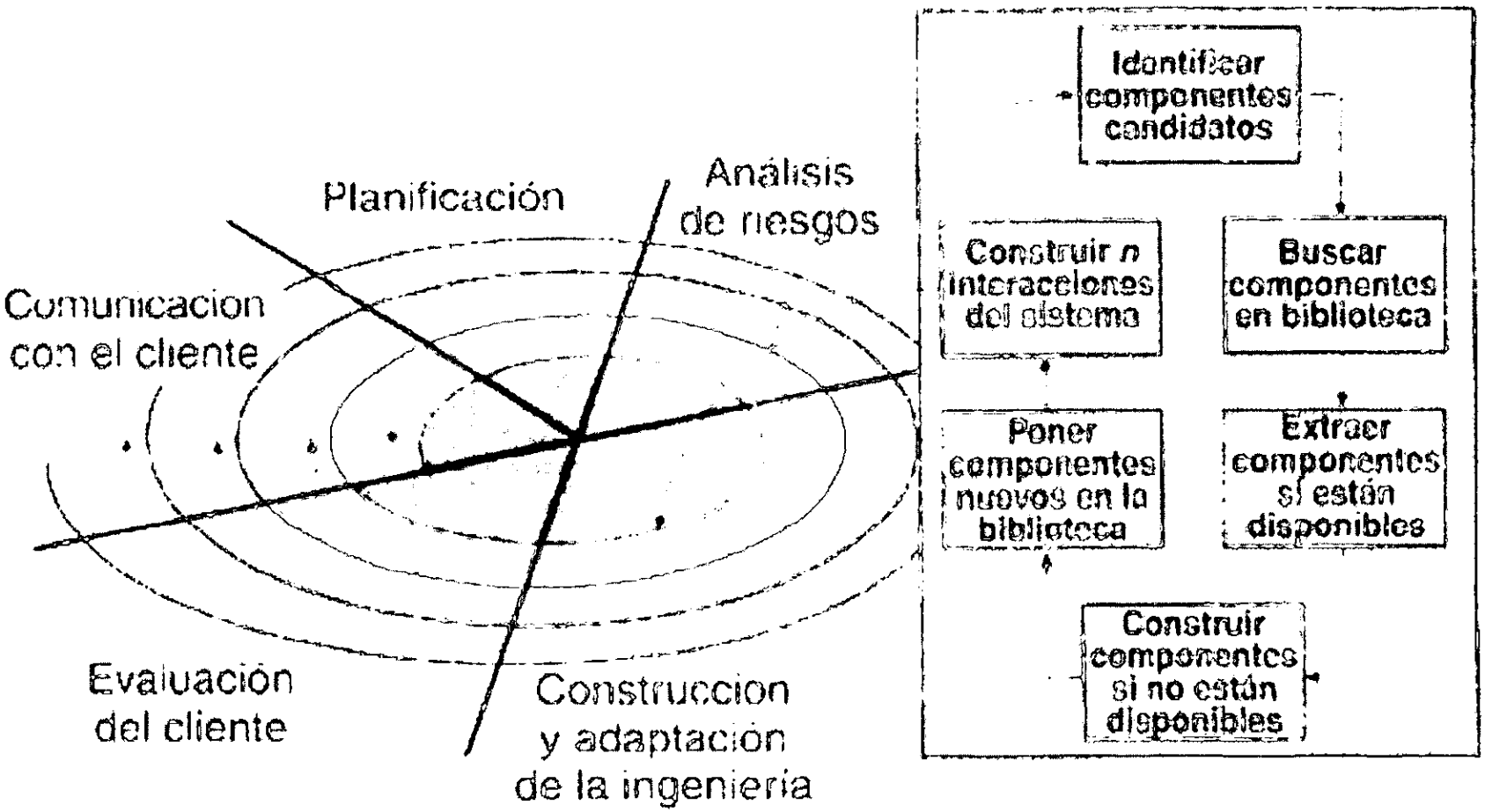
Planificación
Análisis de riesgos
Comunicacion con el cliente
Evaluación del cliente
Construccion y adaptación de la ingeniería
Identificar componentes candidatos
Construir n interacciones del sistema
Buscar componentes en biblioteca
Poner componentes nuevos en la biblioteca
Extraer componentes si están disponibles
Construir componentes si no están disponibles

Figura 18

3.4.7 El modelo de desarrollo concurrente

El desarrollo del software por lo general están dirigidos en acción del tiempo, por lo tanto los modelos de procesos de desarrollo también. Cuanto mas tarde sea, más atrás se encontrará el proceso de desarrollo, un modelo de desarrollo concurrente se basa en las necesidades del usuario, las decisiones de la gestión y los resultados de las revisiones.

El modelo de proceso concurrente se representa en forma de esquema como una serie de actividades importantes, tareas y estados asociados a ellas.

Como podemos apreciar en la FIGURA 19, la actividad se puede encontrar en uno de los estados destacados anteriormente en un momento dado. Todas las actividades existen concurrentemente pero residen en estados diferentes.

Como ejemplo digamos que, al principio del proyecto, la actividad de comunicarnos con el cliente, la cual no es mostrada en la figura, llegó al fin de su primera interacción y existe ya en el estado de *cambios en espera*. La actividad de análisis se encuentra implícita en el estado de *ninguno*, esto es mientras comienza la comunicación con el cliente; ahora se hace la transición al estado de *bajo desarrollo.* Sin embargo, si el cliente indica que debe hacer cambios en requisitos, la actividad de *análisis* cambia del estado bajo desarrollo al estado de cambios en espera.

Un elemento del modelo de proceso concurrente

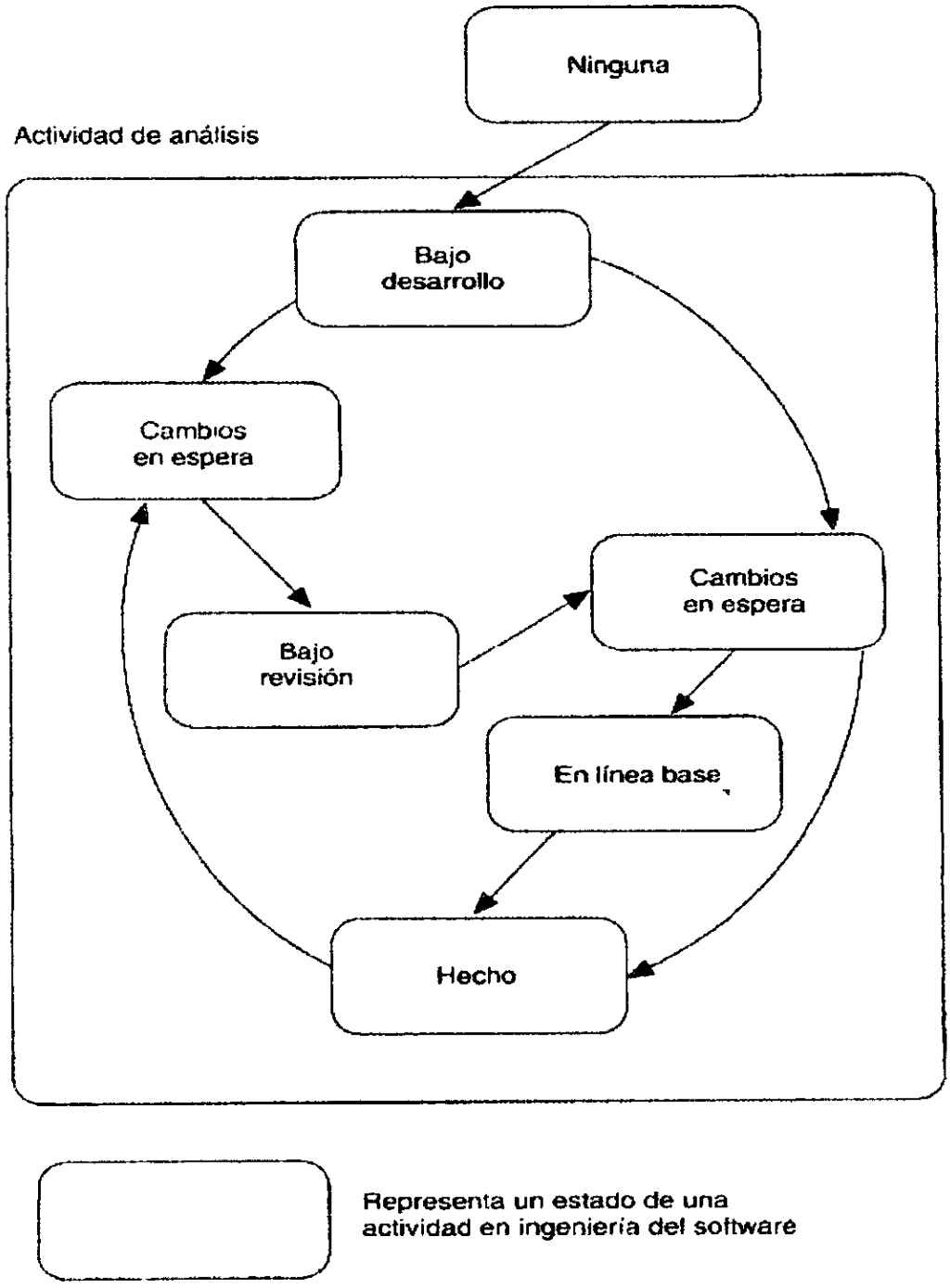

Figura 19

Este modelo es utilizado muy a menudo como el paradigma de desarrollo de aplicaciones de cliente/servidor. Un sistema de cliente/servidor está compuesto por un conjunto de componentes funcionales; aplicado al cliente/servidor, este modelo define actividades en dos dimensiones: la primera, que es la de sistemas, y la otra de componentes. El nivel de sistemas se lleva a cabo en tres actividades denominadas: *diseño, ensamblaje y uso,* mientras que la dimensión de componente se afronta con tan sólo dos actividades denominadas: *diseño y realización.*

La realidad es que este modelo puede ser aplicado a todo tipo de desarrollo de software, proporcionando una imagen exacta del estado actual de un proyecto.

Ventajas:

- Define una red de actividades.
- Todas las actividades de dicha red existen simultáneamente con otras.

Desventaja:

1. En este modelo experimentamos la falta de responsabilidad y compromiso.

3.4.8 Modelo de métodos formales

Los métodos formales permiten al ingeniero de software especificar, desarrollar y verificar un sistema basado en computadora aplicando notaciones rigurosas. Cuando utilizamos los métodos formales somos capaces de formar un mecanismo para eliminar muchos de los problemas que son difíciles de superar con los grandes paradigmas del diseño de software. La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen fácilmente mediante la aplicación de análisis matemáticos. Al usar el modelo de métodos formales durante el ciclo de diseño, sirve como base para la verificación de programas, permitiendo al ingeniero descubrir y corregir errores que no fueron detectados de otra manera.

La Enciclopedia de la ingeniería del software define los métodos formales de la siguiente forma:

> *"Los métodos formales que se utilizan para describir sistemas de computadoras son técnicas de base matemática para describir las propiedades del sistema.*
>
> *Estos métodos formales proporcionan marcos de referencia en el seno de los cuales las personas pueden especificar, desarrollar y verificar los sistemas de manera sistemática, en lugar de hacerlo* ***ad hoc*** *'adecuado'.*
>
> *Se dice que un método es formal si posee una base matemática estable, que normalmente vendrá dada por un lenguaje formal de especificación. Esta base proporciona una forma de definir de manera precisa, notaciones tales como la consistencia y la completitud, y, lo que es aun más relevante, la especificación, la implementación y la corrección."*

Aunque todavía no existe un enfoque establecido, el modelo de métodos formales ofrece la promesa de un software libre de defectos. Sin embargo, se ha hablado de una gran preocupación sobre su aplicación en un entorno de gestión:

1. El desarrollo de los modelos formales actualmente es bastante caro y lleva mucho tiempo.
2. Se requiere un estudio caro porque responsables del desarrollo de software tienen los antecedentes necesarios para aplicar métodos formales.
3. Es difícil utilizar los modelos como un mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos.

El modelo de métodos formales podría tener más demanda en los desarrolladores del software que construyen software de mucha seguridad; tal es el caso del software de los desarrolladores de aviónica y dispositivos médicos, y entre los desarrolladores que pasan grandes prenurias económicas al aparecer errores de software.

CAPITULO IV

EL CONTROL

Qué es el control- Por qué se requiere el control- Principios del control- Reglas de control- Sistemas de control.

4.1 Qué es el control

Podemos decir que el control es el proceso que permite garantizar que las actividades reales se ajusten a las actividades proyectadas anteriormente en la planeación.

El control nos sirve para monitorear la eficacia de nuestras actividades ya sean de planeación, organización y dirección. Una parte esencial del proceso del control consiste en tomar las medidas correctivas que se requieren.

Algunos autores lo definen diferente, tales son los casos de J. Maddok,[17] quien lo define como: "La medición de los resultados actuales y pasados, en relación con los esperados, ya sea total o parcialmente, con el fin de corregir, mejorar y formular nuevos planes." Por su parte, Robert J. Mockler[18] destaca los elementos esenciales del proceso del control: "El control administrativo es un esfuerzo sistemático para establecer normas de desempeño con objetivos de planeación, para diseñar sistemas de retroinformación, para comparar los resultados reales con las normas previamente establecidas, para detectar si existen desviaciones y para medir su importancia, así como para tomar aquellas medidas que se necesiten para garantizar que todos los recursos de la empresa se usen de la manera más eficaz y eficiente posible para alcanzar los objetivos de la empresa"; esta definición divide al control en cuatro pasos esenciales (ver figura 20):

[17] Citado en REYES PONCE, A., obra citada en la nota 7, p. 355.
[18] Citado en STONER, James, obra citada en la nota 4, pp. 610-611.

Pasos básicos del proceso de control

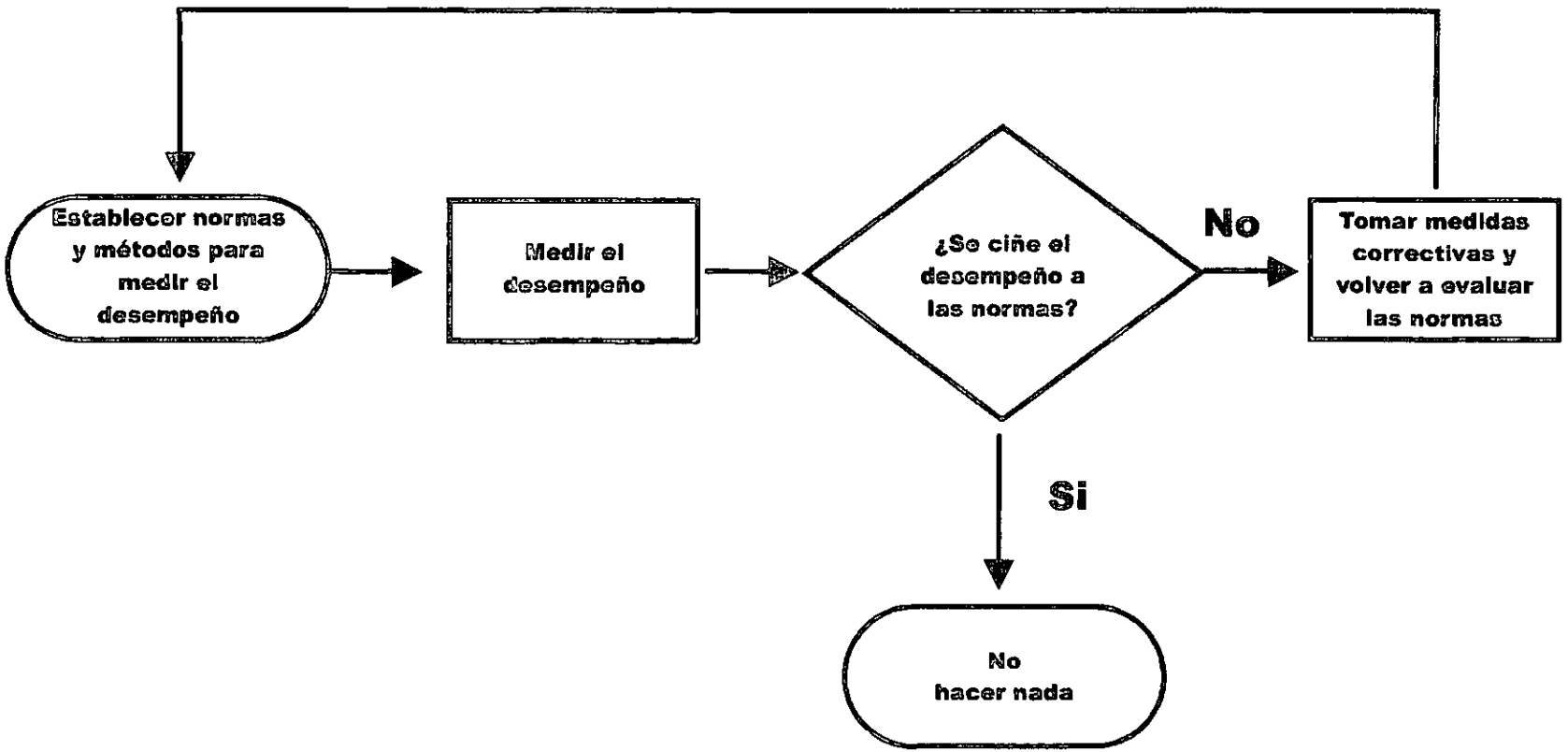

Figura 20

1° Establecer normas y métodos para medir el rendimiento

Idealmente, las metas y objetivos están establecidos en el proceso de planeación definidos en términos claros y mensurables, incluyendo fechas límite específicas. Esto es muy importante, en primer lugar, las metas definidas en forma vaga, por ejemplo, "mejorar las habilidades de los programadores" son igual a palabras huecas, mientras los encargados del desarrollo del sistema, en este caso los ingenieros de software, no empiezan a especificar qué quieren decir con "Mejorar" y lo que pretenden para alcanzar esta meta y cuándo. En segundo lugar, las metas enunciadas con exactitud, por ejemplo, "Mejorar las actividades de los programadores realizando cursos de capacitación semanales en nuestras instalaciones" se pueden medir mejor, en cuanto exactitud y utilidad, que las palabras huecas. Por ultimo, los objetivos mensurables, enunciados con exactitud, se pueden comunicar con facilidad y traducir a normas y métodos[19] que pueden ser utilizados para medir los resultados.

2° Medir los resultados

Al igual que todos los demás aspectos del control, la medición es un proceso constante y repetitivo. La medida que se utilice dependerá del tipo de actividad que se mida. Cabe señalar que los buenos ingenieros del software y los analistas de sistemas suelen evitar que transcurran plazos largos entre las mediciones de los resultados.

3° Determinar si los resultados corresponden a los parámetros

Se pudiera decir que este es el paso más fácil del proceso de control. Las dificultades se han superado en los dos pasos que se presentaron anteriormente; ahora sólo es cuestión de comparar los resultados medidos con las metas o criterios previamente establecidos. Si los resultados corresponden a las normas, los ingenieros del software pueden suponer que "todo está bajo control".

[19] Modo ordenado de proceder para llegar a un resultado o fin determinado, los métodos están compuestos por conjuntos de normas.

4° Tomar medidas correctivas

Este paso sólo será necesario en el momento de darnos cuenta que no se están cumpliendo los estándares establecidos y si el análisis indica que se deben tomar medidas. Estas medidas correctivas pueden involucrar cambios en una o algunas actividades de la organización.

4.2 Por qué se requiere el control

Podemos decir que una de las razones por las que se requiere el control es porque el mejor de los planes se puede desviar. Cabe también mencionar que el control nos sirve para vigilar los cambios de ambiente, así como sus repercusiones en el avance de la organización. Algunos de los cambios ambientales más apremiantes son la índole cambiante de la competencia, la necesidad de acelerar el ciclo de pedidos-entregas.

La importancia del control radica en que es el encargado de cerrar el círculo de toda administración, ya que, como se ha mencionado antes, es el encargado de verificar que todos los objetivos, así como las metas planeadas, se cumplan, y de no ser así, aplicar las medidas correctivas para poder llegar a éstas.

4.3 PRINCIPIOS DEL CONTROL

4.3.1 Del carácter administrativo del control

Es necesario distinguir "las operaciones" de control, de "la función" de control.

La función es exclusivamente de carácter administrativo y es la respuesta al principio de la delegación, la cual no se podría dar sin el control. Cuanta mayor sea la delegación que necesitemos, se requerirá de mayor control. Por ello, podemos decir que el control como

función sólo corresponde al administrador, llamado ingeniero de software para nuestro interés.

En cambio, "las operaciones" son de carácter técnico. Por lo mismo, son un medio auxiliar a la línea en sus funciones, por lo cual deben actuar como "staff". De aquí nace la necesidad de convencer, y no de imponer, los medios de control.

4.3.2 De los estándares

El control es imposible si no existen "estándares"[20] de alguna manera prefijados, y será tanto mejor, cuanto más precisos y cuantitativos sean dichos estándares.

Si el control es comparación de lo realizado con lo esperado, es lógico que, de alguna manera, supone siempre una base de comparación previamente fijada.

Comúnmente esta base se refiere a realizaciones anteriores, meras estimaciones empíricas. Pero no podríamos decir que medimos algo, si lo que estamos obteniendo no lo valorizamos, y, para ello, lo comparamos con algo. De aquí nace la regla de afinar y perfeccionar los estándares, como un medio de preparar el control.

4.3.3 Del principio de la excepción

El control administrativo es mucho más eficaz y rápido cuando se concentra en los casos en que no se logró lo previsto, más bien que en los resultados que se obtuvieron como se había planeado.

Este principio, se enfoca a aprovechar los beneficios que resultan como lo ordinario, el cumplimiento de las previsiones, y las desviaciones que no se pueden evitar, como lo excepcional; hacia estas desviaciones es a donde debemos dirigir toda la atención.

---

[20] Los estandares se pueden definir como patrones uniformes, muy generalizados de alguna cosa.

En realidad, este principio se inclina en convertir el cumplimiento en lo normal, y las desviaciones de los planes en lo excepcional .

4.4 REGLAS DEL CONTROL

Para poder llevar a cabo un buen control en nuestros objetivos y metas, es necesario tomar en cuenta ciertas reglas a las cuales se tendrá que adaptar el control.

Estas reglas son las siguientes:

⇒ **Los controles deben de ser flexibles.** Muchos están en contra del empleo de controles, cuando precisamente éstos no son flexibles, ya que cuando un control no es flexible, un problema que exija rebasar lo calculado en la planeación, hace que, o bien no pueda realizarse adecuadamente la función, o bien se tienda a abandonar el control como inservible.

⇒ **Los controles deben conducir por sí mismos de alguna manera a la acción correctiva.** El control no sólo debe decir que algo está mal, sino dónde, por qué o quién es el responsable.

⇒ **Al utilizar los datos del control, éstos deben de seguir un sistema.** Sus pasos principales serán:

a) Análisis de los hechos.

b) Interpretación de los mismos.

c) Adopción de medidas aconsejables.

d) Su iniciación y revisión estrecha.

e) Registro de los resultados obtenidos.

⇒ **El control puede servir para lo siguiente:**

- Corrección de los defectos.
- Mejoramiento de lo obtenido.
- Nueva planeación general.
- Motivación del personal.

4.5 SISTEMAS DE CONTROL

La planeación temporal de cualquier proyecto de desarrollo de software es muy similar a cualquier proyecto de ingeniería basados en la multitarea, por lo cual podemos hacer uso de las herramientas de la planeación temporal de proyectos.

Se pueden mencionar algunos de los sistemas más importantes que para nuestro fin convienen, como la Técnica de evaluación y revisión de programas (PERT) y el Método del camino crítico (CPM). Estos métodos pueden ser aplicados al desarrollo de software, ya que ambas técnicas se dirigen por la información desarrollada en las actividades anteriores a la planeación del proyecto; asimismo, proporcionan algunas herramientas cuantitativas, las cuales permiten al encargado de la planeación de tareas del desarrollo a:

a) Determinar el camino crítico.

b) Establecer las dimensiones de tiempo más probables para las tareas individuales aplicando modelos estadísticos.

c) Calcula las limitaciones de tiempo que definen una ventana de tiempo de una tarea determinada.

Las técnicas PERT y CPM se han implementado como herramientas fáciles de usar y validas para todos los gestores de proyectos del software.

4.5.1 *Gráficos de tiempo*

Cuando podemos crear una planeación temporal de un proyecto de software, el encargado de la planeación da el primer paso a un conjunto de tareas, mencionando, en primer lugar, la estructura de la descomposición del trabajo a realizar. Si empleamos las herramientas automáticas, dicha descomposición de trabajo se introduce como un esquema de tareas, teniendo como entrada de tarea a la duración, esfuerzo y fecha de inicio, que se le asignan a individuos específicos.

La consecuencia de cada una de las entradas se manifiesta como un gráfico de tiempo denominado Gráfico de Gantt. Podemos generalizar el proyecto y elaborar un solo gráfico o dividirlo para poder desarrollar gráficos para cada una de las funciones del proyecto o para cada individuo implicado en el desarrollo del proyecto.

Debemos el nombre de Gráficas de Gantt a Henry L. Gantt en honor a su nombre. Esta gráfica consiste en representar cada una de las actividades por medio de barras horizontales, las que, por su cruce con los niveles o líneas verticales, indican los días, semanas o meses, etcétera, el momento en que inicia y termina la actividad, y simultáneamente con otras actividades relacionadas con ella. En algunas ocasiones indican también a la persona o sección encargada de cada actividad.

Estas gráficas, aunque son sencillas en su concepto, al principio del siglo se consideraron como instrumentos revolucionarios en la administración, y aún se siguen empleando.

La figura 21 muestra de manera más clara el formato que debe de tener un gráfico.[21] En ella se puede observar una parte de la planeación de un proyecto de software enfatizando la tarea de ámbito de concepto para un nuevo producto de software de procesador de textos. El conjunto de tareas del proyecto aparece en la parte izquierda en forma de una lista. A cada una de las tareas en la lista se le asigna una barra horizontal, que indica la duración de cada una de éstas; en ocasiones aparecen múltiples barras al mismo tiempo en la planeación de las

[21] En PRESSMAN, Roger, obra citada en la nota 1, p. 115.

actividades, lo cual indica una concurrencia en las tareas. Los rombos que apreciamos en la imagen nos indican hitos.

Cuando ya tenemos la información necesaria, podemos generar nuestro gráfico de tiempo; esto nos conduce a la elaboración de tablas de proyecto, en donde encontramos un listado tabular de todas las tareas del proyecto, con fechas previstas, inicio de proyecto y finalización del mismo, como se muestra en la figura 22.[22]

[22] Ibidem, p. 116.

Un ejemplo de gráfico de tiempo

1. Tareas	Semana 1					Semana 2					Semana 3					Semana 4					Semana 5				
1.1.1 Identificar necesidades y beneficios																									
Reunirse con los clientes																									
Identificar las necesidades y las limitaciones del proyecto																									
Establecer la declaración de producto																									
Hito: declaración de producto definida			♦																						
1.1.2 Definir las salidas/control/entradas deseadas (OIC)																									
Alcance de las funciones del teclado																									
Alcance de las funciones de entrada de voz																									
Alcance de los modos de interacción																									
Alcance de documento de diagnósticos																									
Alcance otras funciones WP																									
Documento OCI																									
FTR: revisar el OCI con el cliente																									
Revisar el OCI según se requiera										♦															
Hito: OCI definido																									
1.1.3 Definir la funcionalidad/compartimentación																									
Definir las funciones del teclado																									
Definir las funciones de entrada de voz																									
Describir los modos de interacción																									
Describir las comprobaciones de ortografía/gra/gramática																									
Describir otras funciones WP																									
FrRI Revisar la definición OCI con el cliente																									
Revisar según sea necesario																♦									
Hito: Definición de OCI completa																									
1.1.4 Aislar los elementos software																									
Hito: Elementos software definidos																									
1.1.5 Investigar la disponibilidad del software existente																									
Investigar los componentes de edición de texto																									
Investigar los componentes de la entrada de voz																									
Investigar los componentes de la administración de archivos																									
Investigar los componentes de la comprobación de ortografía y gramática																									
Hito: Componentes reutilizables identificados																	♦								
1.1.6 Definir la viabilidad técnica																									
Evaluar la entrada de voz																									
Evaluar la comprobación de gramática																									
Hito: Viabilidad técnica valorada																						♦			
1.1.7 Hacer una estimación rápida del tamaño																									
1.1.8 Crear una definición de ámbito																									
Revisar el documento de ámbito con el cliente																									
Revisar el documento según se requiera																									
Hito: Documento de ámbito completado																									♦

Figura 21

Un ejemplo de tabla de proyecto

Tareas de trabajo		Inicio previsto	Inicio real	Terminación prevista	Terminación real	Personas Asignadas	Esfuerzo	Observaciones
1.1.1	Identificar necesidades y beneficios							
	Reunirse con los clientes	Sem 1,dl	Sem 1,dl	Sem 1,d2	Sem 1,d2	BLS	2 p-d	Es previsible que
	Identificar las necesidades y limitaciones de¡ proyecto	Sem 1,d2	Sem 1,d2	Sem 1,d2	Sem 1,d2	jpp	1 p-d	requiera más
	Establecer la declaración de producto	Sem 1,d3	Sem 1,d3	Sem 1,d3	Sem 1,d3	BLSIJPP	1 p-d	esfuerzo/tiempo
	Hito: declaración de producto definida	Sem 1,d3	Sem 1,d3	Sem 1,d3	Sem 1,d3			
1.1.2	Definir las salidas/controllentradas deseadas (OCI)							
	Ámbito de las funciones del teclado	Sem 1,d4	Sem 1,d4	Sem 2,d2		BLS	1,5 p-d	
	Ámbito de las funciones do entrada de voz	Sem 1,d3	Sem 1,d3	Sem 2,d2		jpp	2 p-d	
	Ámbito de los modos de interacción	Sem 2,dl		Sem 2,d3		MLL	1 p-d	
	Ámbito de los diagnósticos del documento	Sem 2,dl		Sem 2,d2		BLS	1,5 p-d	
	Ámbito de otras funciones WP	Sem 1,d4	Sem 1,d4	Sem 2,d3		jpp	2 p-d	
	Documento OCI	Sem 2,dl		Sem 2,d3		Mil	3 p-d	
	FRT-. revisar OCI con el cliente	Sem 2,d3		Sem 2,d3		Todos	3 p-d	
	Revisar OCI según se requiera;	Sem 2,d4		Sem 2,d4		Todos	3 p-d	
	Hito: OCI definido	Sem 2,d5		Sem 2,d5				
1.1.3	Definir la funcionalidad/comportamiento							

Figura 22

El gestor de proyecto puede hacer un seguimiento del progreso al emplear las tablas con los gráficos de tiempo.

La planeación del proyecto proporciona a los gestores las guías necesarias, "llamadas mapas de la carretera". Cuando se han desarrollado satisfactoriamente, las tareas e hito se definen y son controlados a medida que el proyecto avanza.

Este desarrollo o seguimiento se puede llevar a cabo de diferentes formas:

- Podemos realizar juntas periódicas donde se informara el estado del proyecto, todos los miembros del equipo deberán dar a conocer el progreso y sus inconvenientes.
- Comparando las fechas reales de inicio con las planeadas para cada una de las tareas de todo el proyecto, éstas estarán listadas en la tabla del proyecto
- Se pueden evaluar los resultados de todas las revisiones que se realizaron conforme el avance del proceso de ingeniería del software.
- Determinando si se han conseguido los hitos formales del proyecto.
- Se pueden obtener valoraciones subjetivas del progreso hasta la fecha y los problemas que se aproximan, éstos son proporcionados por los desarrolladores del software.

Todas estas técnicas de seguimiento son utilizadas por los gestores experimentados. El control es utilizado para administrar los recursos del proyecto. Si todo el proyecto no presenta inconvenientes, podemos utilizar un control menos riguroso; pero si aparece un problema, el gestor es el encargado de solucionarlo a la brevedad posible.

Cuando existe una fecha de entrega muy precipitada, podemos utilizar una planeación del proyecto junto con una técnica de control denominada *time boxing* o *tiempo encajonado.* Al utilizar esta técnica estamos aceptando que no podemos entregar el producto completo en la fecha establecida, por lo cual recurrimos a los paradigmas incrementales, creando una planeación temporal para cada una de las entregas.

A lo que se refiere el *time boxing* es a empaquetar las tareas y los incrementos en el tiempo, haciendo que las tareas se ajusten hacia atrás desde la fecha límite de entrega para

cada uno de los incrementos. El procedimiento es muy sencillo, ya que sólo basta con poner una caja con cada tarea. Al alcanzar el límite de la caja de tiempo para la tarea, que aproximadamente es del 10%, se termina el trabajo y se prosigue con la siguiente tarea.

4.5.2 *PERT*

La técnica PERT[23] recibe el nombre de las siguientes siglas: Program Evaluation and Review Technique (Técnica de Evaluación y Revisión de Programas). Consiste en un instrumento en el que, con base en una red de actividades y eventos, y mediante la estimación de tres tiempos, se evalúa la probabilidad de terminar un proyecto para una fecha determinada.

Esta técnica fue creada por Willard Fazard en 1958, quien pertenecía a la oficina de proyectos espaciales de la oficina naval de ordenanza de la marina norteamericana, con el objetivo de controlar el proyecto de lanzamiento del proyectil "Polaris" de la NASA.

Aunque esta técnica fue creada inicialmente para controlar y evaluar la duración de proyectos, por lo que se le conoció como PERT/tiempo, posteriormente se han introducido en ella los costos de las actividades, para efecto de control presupuestal, y aun para estudiar el tiempo mínimo compatible con el menor costo posible, dando lugar al sistema PERT/costo. Con todo, esto último suele medirse más eficientemente con la técnica CPM que describiremos en la siguiente sección.

Operación de la técnica PERT

Los pasos que comprende son esencialmente los siguientes:

[23] En REYES PONCE, A., obra citada en la nota 7, pp. 370-375.

1. Formación de una lista de las actividades que integran el proyecto

Lo primero que debemos hacer es que las personas que conocen con todo el detalle el trabajo a desarrollar, en este caso pueden ser los analistas del software, formulen una lista en la que anoten todas las actividades que se requiere llevar a cabo, sin importar el orden en que se consignen. Se entiende por actividad, a todo trabajo, de cualquier orden que sea, manual, intelectual, administrativo, etcétera, que habrá de requerir determinado tiempo para su ejecución, representando determinados costo, porque consume recursos materiales y humanos. Toda actividad la representaremos con una flecha.

2. Determinación de la secuencia u orden de las actividades

Este segundo paso consiste en precisar, respecto de cada actividad, cuál debe de ir primero, y cuál debe de ir después. La primera constituye un requisito previo de la que se esté considerando, mientras que la segunda no puede darse si no se ha terminado la que se está ordenando. Pueden existir actividades simultáneas, o sea, aquellas que no constituyen un requisito anterior, ni posterior, respecto a la que se está tomando en cuenta para su ordenación.

El medio más práctico y fácil para determinar esta secuencia consiste, quizá, en usar tarjetas, en cada una de las cuales se anota una actividad, las que después se colocan en el orden adecuado, anotando números cuando están ordenadas, cuidando de poner el mismo numero a las actividades simultáneas.

3. Trazo de la red

De los tres métodos de trayectoria crítica, el trazo de la red es lo más típico. Una red es un diagrama o gráfica integrada por flechas que representan las actividades, y por círculos, que representan los eventos, unidos en forma de poder indicar su relación de dependencia cronológica y de secuencia. Los eventos son "puntos identificables en el tiempo, en los que ha acontecido, o alguna situación se ha originado", según la definición de Codier "Son los hechos administrativos que dan origen a una actividad". Cuando varias flechas concurren en el mismo evento o arrancan de él, se le llama Nodo.

A continuación, para explicar el trazo de una red, se dan las reglas respectivas:

a) Cada actividad tiene en el diagrama un evento por punto de origen, y otro por punto de terminación.

DIAGRAMA DE EVENTOS

FIGURA 23

a) Los eventos deben numerarse, procurando, en lo posible, que los números sigan el orden mismo que las actividades que limitan dentro del proceso.

b) Las actividades se identifican mediante los números de los eventos en los que comienzan y terminan: así, la actividad 6-9 es una actividad que comenzó en el evento 6 y terminó en el evento 9. Representado en la siguiente figura.

REPRESENTACIÓN DE INICIO Y TERMINO DE UNA ACTIVIDAD

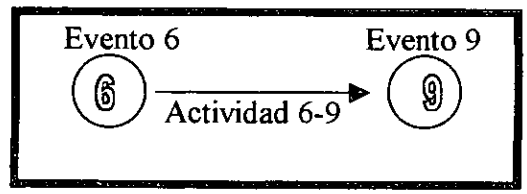

FIGURA 24

c) De un mismo evento pueden salir una o varias actividades, y a un mismo evento pueden llegar varias de ellas, con tal de que las flechas no salgan de la mitad de la actividad, ya que no se puede dar comienzo a una actividad, hasta terminar por completo la inmediata anterior. Si una actividad se puede iniciar antes de que la anterior esté terminada, se indica el % o tiempo de duración que requiere, pero indicando de algún modo que las dos partes de % o del tiempo forman la misma actividad.

A partir de una actividad sale una o más actividades

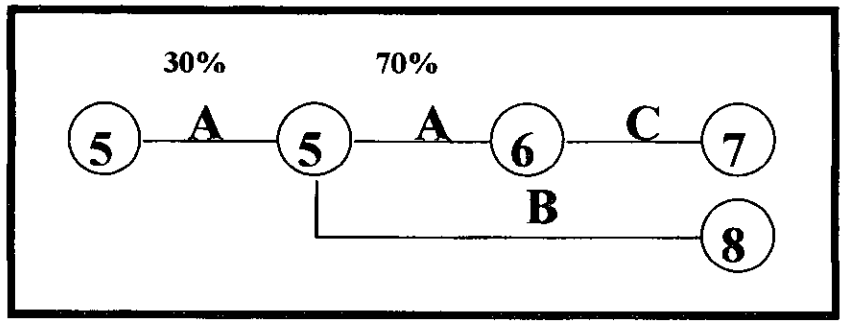

FIGURA 25

d) Las flechas pueden ser ascendentes o descendentes, trazarse en líneas continuas o discontinuas, etcétera, para ayudarnos a hacer indicaciones y distinciones. Lo segundo suele usarse para indicar sólo que un evento tiene relación con otros, sin que exista dependencia, lo cual impide que un evento quede suelto, lo que haría la malla incompleta. También para distinguir las rutas mayores de la trayectoria, o ruta crítica, que se marca siempre con línea llena y aún más gruesa. Se llama ruta crítica el conjunto de actividades cuya duración resulta más larga, porque, sin importar que las actividades simultáneas a ella puedan ser más cortas, lo que condicionará la duración de todo el programa será *la ruta critica.*

Beneficios de la técnica PERT

1. Ante todo, nos permite conocer las fases criticas de un proyecto, señalando de esa manera aquellas actividades que deben cuidarse con máxima atención, ya que, al alargarse ellas, aunque se acorten todas las demás, todo el proyecto se alargaría.
2. Permite analizar cómo pueden reducirse las rutas criticas, con lo que todo el proyecto puede hacerse menor.
3. Constituye una representación gráfica de todo un proyecto, con su estimación de tiempos y relaciones, muy apta para explicar a todos el momento en que se encuentra su desarrollo.
4. Permite tener un esquema, no sólo de la magnitud del trabajo, sino de cada una de sus partes y, sobre todo, de su interrelación, lo cual nos permiten métodos como las Gráficas de Gantt.
5. Pueden emplearse en toda clase de programas, ya que todo lo estudia sobre la base de un mismo patrón: actividades, estimadas en el tiempo que cada una quiere, y eventos o nodos, que se consideran atemporales. Así, es aplicable en programas de mercadotecnia, instalación de maquinaria, desarrollo de programas de adiestramiento de personal, realización de pedidos especiales, y para nuestro fin común el desarrollo de software.
6. Fuerza a los encargados del proyecto a planear, ya que es importante desarrollarla sin afinar de algún modo los planes.
7. Nos permite ver a la mitad de nuestros proyectos, si lo pasado se ha cumplido, y, en caso de haberse retrasado u omitido algo, tratar de recobrar tiempos, reconstruir lo que falto, etcétera.

Limitaciones del PERT

Tiene, como se comprende, evidentes limitaciones:

a) Así, para operaciones rutinarias no tiene aplicación, sino más bien sirve para procesos que son de suyo únicos y no repetitivos.

b) Cuando la mayor parte de los eventos están ligados por actividades sucesivas, que producen un diagrama lineal, es también inaplicable.

c) Lo mismo ocurre cuando es imposible o muy aleatorio el cálculo de tiempos para cada actividad.

A este último respecto, debe advertirse que el PERT trabaja con tres tipos de tiempos: Tiempo óptimo (to), Tiempo normal (tn), y Tiempo pesimista (tp). En forma estadística suele establecerse el tiempo estimado por la combinación de estos tres tiempos, mediante la medición de la varianza respectiva.

4.5.3 CPM

Simultáneamente con el estudio del método PERT, aunque en forma independiente, las compañías Dupont Nemours y Remington Rand, a través de sus técnicos Morgan R. Walker y James E. Kelly Jr., buscaban un procedimiento que les permitiera resolver problemas típicos de programación. Llegaron al resultado de redes de actividades, como en el caso del PERT, por lo que la primera fase del CPM (Critical Path Method <Método de la Ruta Crítica>) es prácticamente igual al PERT, del cual difiere porque trabaja solamente con un equipo probable de ejecución, basado en experiencias previamente registradas, pero a la vez introduce costos estimados de las actividades implicadas en el proyecto, buscando acortar el proyecto al condensar ciertos tiempos, para lograr un mínimo costo. Podríamos definirla, como la técnica que estima un tiempo probable y determina el costo de cada actividad de una

red, con el fin de fijar el tiempo más conveniente de acortamiento en la duración de un proyecto, para lograr el mínimo coste posible.

CONCLUSIONES

De acuerdo al estudio realizado me percate que dentro del desarrollo del software existe un problema muy marcado, el cual no se puede dejar pasar por alto, ya que de este depende la vida de nuestro producto. Al evitar este problema, nos aseguramos de obtener sistemas con CALIDAD.

La ingeniería del software está introduciéndose a la cuarta década de su existencia. Sufriendo cambios muy radicales llegando a una etapa media con muchos logros pero con un trabajo de desarrollo muy significativo a futuro. Así mismo, ha desplazado al programador y lo ha sustituido por el "ingeniero de software", quien trata de utilizar los diferentes modelos y herramientas existentes en el medio, con el fin de poder desarrollar un producto de calidad, el cual asegure el óptimo funcionamiento para lo cual fue creado.

El desarrollo de software depende de la utilización de los modelos de la planeación, pero a ciencia cierta no podemos recomendar un modelo especifico para el desarrollo, ya que esto depende de diferentes factores como lo son : el tiempo, la capacidad del ingeniero y las necesidades del cliente.

El estado de ingeniería del software sigue siendo un estudio muy diversificado ; las actividades han cambiado, obteniendo un adelanto razonable y quedando mucho por hacer antes de que esta disciplina alcance su punto más alto.

BIBLIOGRAFÍA

BROOKS, F., ***The Mythical Man-Month,*** Addison Wesly, 1975.

CENCADE, ***Manual del participante. Q.F.D. Servicio,*** CENCADE, México, 1996.

FAIRLEY, Richard, ***Ingeniería del software,*** McGraw-Hill, México, 1998.

LIENTZ, B., ***Software Maintenance Management,*** Addison Wesley, 1980.

PRESSMAN, Roger S., ***Ingeniería del software,*** 4ª ed., México, McGraw-Hill, 1998.

REYES PONCE, Agustín, ***Administrción de empresas (primera y segunda parte),***

Editorial Limusa, México, 1993.

STONER, James A.F., ***Administración,*** 6ª ed., Prentice-Hall, México, 1996.