12
24

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

CALIDAD EN EL DESARROLLO DE SISTEMAS

SEMINARIO DE INVESTIGACION
I N F O R M A T I C A

QUE EN OPCION AL GRADO DE:
LICENCIADO EN INFORMATICA
P R E S E N T A N :
ANGELICA MEJIA DE GYVES
JOSE EDUARDO YAÑEZ ORTIZ

PROFESOR DE SEMINARIO
M.B.A. JOSÉ ANTONIO ECHENIQUE GARCÍA

MEXICO, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTROL DE LA CALIDAD EN EL DESARROLLO DE SISTEMAS

INDICE

CAPITULO 3

APLICACION DEL CONTROL DE CALIDAD EN
EL DESARROLLO DE SISTEMAS.

CAPITULO 4

EFECTOS ECONOMICOS DEL CONTROL DE CALIDAD EN EL DESARROLLO DE SISTEMAS

CAPITULO 5

TECNICAS DE VERIFICACION Y VALIDACION

INTRODUCCION

En la actualidad nuestro país se enfrenta ante una inminente apertura económica a nivel internacional, compitiendo con países que poseen esquemas bien definidos de calidad, buscando en primera instancia el tratar de ser más competitivo, pero descuidando las necesidades reales del consumidor final. Esto ha provocado que las empresas mexicanas adquieran sistemas de cómputo de tecnología avanzada, creando en ellas la necesidad de tener áreas de desarrollo de software, o bien la adquisición de sistemas aplicativos que sean compatibles con sus requerimientos de manejo de información. Por ello es de vital importancia la búsqueda constante del mejoramiento de la calidad para obtener como consecuencia y en forma conjunta una auténtica y mayor productividad. La manera de conducir a las empresas hacia la obtención de la calidad, se logra mediante la concientización, es decir a través de la inducción, capacitación y adiestramiento de todos los integrantes de la institución así como también haciendo uso efectivo de las técnicas y herramientas para la correcta toma de decisiones logrando así el control y mejoramiento de la calidad.

Es necesario para nuestro país retomar, actualizar y reorganizar las metodologías usadas para el desarrollo de sistemas incorporando para tal efecto, el concepto de calidad total, el cual significa hacer las cosas bien desde la primera vez en todo lo que nos propongamos hacer, lo cual implica cero defectos. Por ello es importante saber por qué, cómo y donde aplicar la calidad considerando los atributos de la misma que debe reunir un producto y / o servicio de software, así como los factores que influyen en la calidad y productividad de una organización en el proceso de desarrollo de sistemas.

Dada la gran cantidad de prestadores de servicios de cómputo, la competencia entre éstos se ha incrementado, de tal suerte que es muy importante considerar la adopción del concepto de calidad en la elaboración de productos y prestación de servicios, lo cual implica adecuarse a las necesidades específicas del usuario

El concepto de control de calidad exige la participación de todos los miembros de la organización, por esto debemos tomar en cuenta que para lograr un alto nivel de calidad se requiere contar con el personal idóneo, lo cual implica que éste cuente con la capacitación adecuada, así como una constante motivación que propicie un desarrollo integral de manera oportuna.

La interrelación que existe en todas las áreas de la empresa, genera la necesidad de que el personal cuente con una conciencia de calidad para contribuir al desarrollo de productos con calidad.

Adicionalmente, mediante una adecuada estructura organizacional y aplicando una correcta metodología de Ingeniería de Software con políticas, estándares y técnicas bien definidas de verificación y validación, se pueden lograr desarrollos eficientes, oportunos y cuyo presupuesto sea aceptable.

Anteriormente el enfoque de costos para el desarrollo de sistemas era de poca importancia dentro de las organizaciones, en la actualidad es de los elementos que tiene mayor impacto en la realización de un proyecto.

Una adecuada estimación de costos es una herramienta importante aunque no definitiva, que nos brinda un panorama más realista del costo de un proyecto, que de hacerse de manera ineficiente podría convertir al proyecto en incosteable.

Un error que se comete frecuentemente es el pensar que al implantar la calidad en las empresas aseguramos su correcta aplicación, sin embargo es necesario establecer controles que garanticen que en cada una de las etapas de desarrollo de un producto se esta cubriendo con los requerimientos de diseño, así como la realización de pruebas modulares e integrales del sistema traerá como consecuencia la obtención de un sistema terminado acorde con las necesidades y especificaciones de un cliente o usuario.

El principal objetivo de esta tesis es dar una idea de la importancia que tiene la comprensión del concepto de calidad en una organización y todas sus implicaciones, para lograr un adecuado desarrollo de sistemas.

TESIS

LA CALIDAD EN EL DESARROLLO DE SISTEMAS

Importancia de la calidad en el desarrollo de sistemas

UNAM
FCA

EL CAMINO HACIA EL CONTROL TOTAL Y MEJORAMIENTO DE LA CALIDAD

1910	FEDERIC TAYLOR * ADMINISTRACION CIENTIFICA.	TEORIA "X" * HABILIDAD DE LA PERSONA. * GERENCIA NO PARTICIPATIVA. * SOLO HACER EL TRABAJO.
1924	WALTER SHEWART * CONTROL ESTADISTICO DE CALIDAD.	
1930	DOUGLAS MCGREGOR ELTON MAYO * HUMANIZACION DEL TRABAJO.	TEORIA "Y" * SENTIMIENTOS DE LA PERSONA. * QUE OPINE Y PARTICIPE.
1950	EDWARDS DEMING (JAPON) * CONTROL ESTADISTICO DE CALIDAD. EN TODO EL CICLO DE MANUFACTURA.	
1954	JOSEPH JURAN * ADMINISTRACION DE LA CALIDAD.	
1957	HERZGERG, MASLOW Y OTROS * MOTIVACION.	
1950 — 1980	KAORU ISHIKAWA Y OTROS * CONTROL DE CALIDAD AMPLIO EMPRESARIAL. * CIRCULOS DE CALIDAD.	TEORIA "Z" * LA PERSONA QUIERE HACER LAS COSAS BIEN. * GERENCIA PARTICIPATIVA. * AUTOCONTROL.
1960 — 1980	A. FEIGENBAUM * CONTROL TOTAL DE CALIDAD. * COSTOS DE CALIDAD.	

IMPORTANCIA DE LA CALIDAD

Introducir el concepto de calidad en todas y cada una de las actividades que se realizan dentro de una empresa es de vital importancia para la existencia de la misma, no importando el giro de esta. Esto se debe primordialmente a los constantes cambios tan drásticos en los entornos económicos, políticos y sociales que sufre el país. Tan fuerte ha sido en México el impacto de estos cambios a nivel mundial y la necesidad de no mantenerse al margen, que en el año de 1989 la calidad adquiere, por decreto presidencial, el rango de *"valor y prioridad nacional"*.

Por esta razón surgió el Premio Nacional de Calidad, mediante el cual se pretende impulsar y promover una cultura de calidad total y, de este modo, lograr ser más competitivos, elevando la productividad y propiciando con ello el desarrollo de los recursos humanos.

1.1 MEDIDA DE LA NO CALIDAD

1.1.1 DEVOLUCION POR PARTE DEL USUARIO

Cuando un sistema no es funcional durante su fase de liberación en producción ocasiona gastos que son fácilmente evaluables:

- pérdidas de tiempo.
- costo debido a la reprogramación o modificación del sistema.
- pérdida de producción para el usuario.

En este caso, se puede evaluar el costo de la no calidad. La calidad se mide considerando los precios de cumplimiento e incumplimiento de los requisitos:

Costo de la calidad = PDI + PDC

Donde:

PDI= Precio del incumplimiento de los requisitos, es decir, lo que cuesta hacer las cosas mal.

PDC= Precio del Cumplimiento de los requisitos, o sea, lo que se necesita invertir para que las cosas se hagan bien.

El costo de la no calidad comprende lo que son los reprocesos y los desperdicios. La Fig. 1.1 muestra el ciclo de la no calidad. Por ello debemos hacer las cosas bien desde la primera vez y no descansar hasta encontrar la falta real o potencial de la no calidad, en el desarrollo de los proyectos que nos sean encomendados.

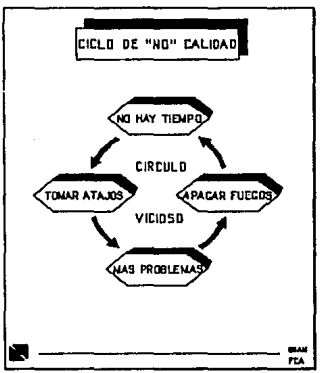


Fig. 1.1

En la medida que no cumplamos con las necesidades de la naturaleza no podremos cubrir con la calidad hacia el consumidor final (usuario, cliente).
La calidad total empieza por la gente (recursos humanos), no por las cosas, sin embargo se refleja en las cosas que son producidas por la gente.

1.2 CICLO DE LA CALIDAD

Para lograr y asegurar la calidad al usuario o cliente, es necesario controlar la calidad en cada una de las siguientes etapas, desde el diseño del producto o sistema hasta el servicio post-venta. Estas etapas se constituyen como el ciclo de la calidad, habiendo interdependencia y retroalimentación en cada una de ellas.

De acuerdo al ciclo de la calidad (Fig. 1.2), ésta se constituye en el objetivo común para toda la organización y en la forma de administrar la empresa.

Para satisfacer las necesidades o requerimientos del usuario o consumidor, debe existir calidad en:

- El diseño del producto informático.
- Abastecimiento de herramientas propias para desarrollo del sistema.
- Producción.
- Entrega del sistema terminado.
- Venta y/o servicio.
- Administración general.

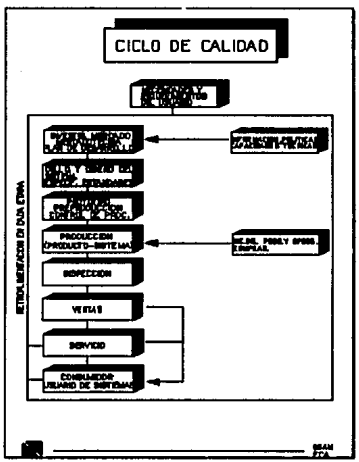


FIG. 1.2

1.3 OBJETIVO DE LA CALIDAD

El uso de la computadora en la era actual y la evolución de la misma de una manera tan vertiginosa han propiciado una gran problemática que obstaculiza la optimización del uso del recurso tan poderoso del cual se dispone. Consecuencia de ello, es lo que se conoce como la "*Crisis del Software*" que consiste en el cumplimiento deficiente de los requisitos del usuario, ocasionado en buena medida por el empleo de erróneas técnicas en el control de proyectos, obteniendo productos a un costo muy elevado y fechas de entrega posteriores a las programadas previamente con el usuario.

Así es como surge una nueva rama de la Ingeniería conocida como Ingeniería de Software que tiene como objetivos primordiales: el incremento de la productividad y satisfacción al trabajo de los profesionistas afines al campo de la computación, mejorar la calidad de los productos del software, proveer de técnicas apropiadas para automatizar el manejo de datos, planear eficazmente los sistemas, documentar, organizar, validar y controlar formalmente la calidad del trabajo realizado, cumplir con los objetivos de la empresa en cuanto a la productividad de sus sistemas computo.

Teniendo como objetivo la calidad, se logra obtener productos con:

- calidad justa como base para ofrecer al usuario el software que satisface plenamente sus necesidades,
- el precio justo,
- el tiempo justo para las fechas de entrega.

La Ingeniería de Software pretende proveer de técnicas apropiadas para automatizar el manejo de datos, planear eficazmente los sistemas, documentar formalmente, organizar, validar y controlar la calidad del trabajo efectuado, así como cumplir con los objetivos de la empresa en relación a la productividad de sus sistemas de computo.

La estrategia es la búsqueda constante del mejoramiento de la calidad para obtener en consecuencia y en forma conjunta una autentica y mayor productividad, la manera de administrar las entidades para obtener calidad y en

el uso efectivo de las técnicas y herramientas; como los métodos estadisticos para la correcta toma de decisiones para el control y mejora de la calidad en el desarrollo del software. Todo esto nos lleva a tener una mayor competitividad.

1.4 CONCEPTO DE LA CALIDAD

El concepto de la calidad se ha desarrollado conjuntamente con el desarrollo de la empresa, debido principalmente a la evolución de los mercados, pasando de ser un mercado del vendedor a un mercado del comprador; por la competencia en aumento, saturación de mercados, desarrollo tecnológico, competencia internacional, desarrollo del consumidor, etc. Contribuyendo todo ello a hacer de la calidad un objetivo estratégico de negocios. El uso mas exacto y práctico del término calidad, es que la calidad sea satisfactoria para el propósito que se intenta, que en términos del precio sea aceptable para el producto o servicio al cual se aplica, y que sea de un nivel que proporcione resultados de confianza; es decir, que el producto o servicio siempre satisfaga la necesidad respecto a la calidad.

Existen diversos conceptos de Calidad:

Calidad es el grado en el que un producto o servicio satisface las necesidades o requerimientos del consumidor-usuario (definición básica), lo cual implica el hacer las cosas necesarias; bien o la primera vez (Calidad en el trabajo); constituyéndose como la meta en cualquier actividad humana, individual o de grupo.

Calidad es la satisfacción de las necesidades apreciada por el cliente o el usuario. Para ello debemos lograr cumplir con las expectativas y requisitos dinámicos (reales) de nuestros clientes -tanto externos como internos-.

De hecho, bajo esta idea se esconde un conjunto de puntos que deben dominarse:

- La fiabilidad del producto o servicio.
- Sus caracteristica y presentaciones.
- El ciclo de vida del sistema.
- La seguridad de datos.

- El costo del desarrollo.

Otra definición de calidad es: *satisfacer las necesidades del usuario considerando:*

a) La relación externa de la empresa.
b) La relación interna.

En otras palabras: *"el siguiente proceso es el usuario".*

De acuerdo con lo anterior, *la calidad es la condición mas importante para lograr la eficiencia, para lograr el trabajo y para mejorar a su vez la productividad.*

La calidad se define como: *la actitud para poder cumplir siempre con los requisitos que requiere satisfacer nuestro cliente, no como excelencia.*

Simplemente, entregar al usuario lo que prometimos, para ello es indispensable considerar el no corregir después de los hechos, es decir, todas las actividades de la organización deben estar orientadas hacia la prevención (ver Fig. 1.3). La norma personal de realización es cero defectos, lo cual no significa ser perfectos sino, el que podamos responder a dos preguntas:

- Con cuántos de los requisitos a los que nos comprometimos debemos cumplir? Con todos.
- Cuántas veces? Siempre.

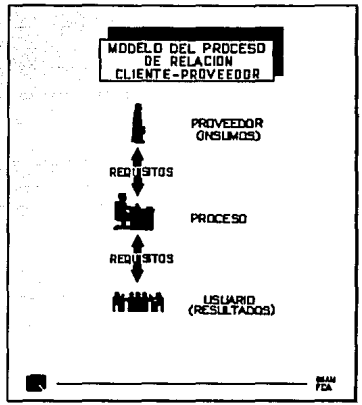


Fig. 1.3

1.5 COMO LOGRAR LA CALIDAD?

Si se quiere alcanzar el concepto de calidad total, o sea, "calidad en todo", tendremos que añadir la aptitud a la satisfacción por parte de los elementos que intervienen en el proceso:

- La satisfacción de la alta dirección.
- La satisfacción del personal involucrado en el desarrollo del sistema.
- La satisfacción del usuario final.

Así mismo, se requiere de un proceso de cambio, el cual debe tener el firme y decidido propósito de mejorar día a día los productos y servicios

que ofrecemos a nuestros clientes (usuarios), logrando con ello su plena satisfacción y el hecho de que podamos ser más competitivos, la exigencia de ser competitivos y alcanzar la calidad total es un gran reto. Sabemos que nadie cambia por cambiar y que todo cambio obedece a una necesidad.

Para lograr la calidad total, se requiere un cambio en la cultura de la organización, mediante la inclusión de nuevos valores a través de un proceso de mejoramiento sistemático y continuo, cambio de cultura, pero sin perder la identidad de la empresa, país, etc.

Esto significa pasar:

De:	A:
* Reactiva	* Proactiva
* Corto plazo	* Largo plazo
* Síndrome de soluciones rápidas.	* Planeación y prevención.

1.6 TIPOS DE CALIDAD EN LOS SISTEMAS

1.- Calidad de diseño.- Son especificaciones, características, programas específicos, procedimientos, etc. que prometen producir un servicio vendible que el cliente (usuario) requiere, calidad de servicio esperada, implica el servicio mas útil.

2.- Calidad de conformancia (trabajo).- Es la medida de la eficiencia en el logro de resultados por la calidad esperada, prometida. Calidad de servicio resultante implica mayor confiabilidad del trabajo.

1.7 APLICACION DEL CONTROL DE CALIDAD Y SU INFLUENCIA EN EL DESARROLLO DE SISTEMAS

El concepto de calidad total en el desarrollo de sistemas debe empezar desde el seno de la empresa, en cada uno de los integrantes de la misma pasando por

cada una de las actividades que involucran el desarrollo de un producto, con el firme propósito de satisfacer al usuario en primera instancia, además de la plena satisfacción del personal involucrado en la empresa.

En resumen, para alcanzar esta meta es necesario que todos promuevan y participen en el control de calidad incluyendo a los altos ejecutivos así como a todas las divisiones de la empresa y a todos los empleados de la misma.

Al aplicar control de calidad en el desarrollo de proyectos de software se solucionan problemas clasificados en los cuatro siguientes puntos:

1.- En las empresas y usuarios en particular.
2.- En la calidad en general.
3.- En la administración de proyectos de software.
4.- En cada una de las fases del ciclo de vida del sistema.

Para considerar un software como un producto de alta calidad se deben establecer:

- Normas mínimas a cumplir.
- Procedimientos en el desarrollo y el control en cada fase del ciclo de vida del producto.
- Estructura organizacional del proyecto.
- Tareas y responsabilidades específicas del personal encargado de llevar a cabo las pruebas.
- Documentación a preparar para revisar la consistencia del producto.
- Técnicas para llevar a cabo auditorías y pruebas requeridas.
- Estándares, normas y especificaciones a usarse.
- Criterios de aceptación del producto.

Las personas encargadas de llevar a cabo el trabajo anterior deberán ser aquellas que conozcan profundamente cada una de las partes del proyecto tanto teórico como práctico. Los únicos que reúnen el perfil para desarrollar este trabajo son los ingenieros de software en sus diversas modalidades así como los

licenciados en informática, ya que combinan actividades técnicas y administrativas.

Dos conceptos muy importantes son inherentes al control de calidad:

a) Garantía de calidad: es la forma en que se asegura la calidad de un producto o servicio, mediante la prevención de errores, este enfoque significa controlar la calidad durante el proceso, o sea durante la realización del trabajo y así asegurar la calidad del producto y/o servicio (resultado). Con este enfoque se logra un beneficio económico mayor.

b) Círculos de calidad: son grupos voluntarios de individuos cuyo objetivo es lograr la calidad y el desarrollo personal. La preparación de un plan de control de calidad del software para cada proyecto es una de las principales responsabilidades del grupo de control de calidad.

Para lograr la calidad y productividad en forma conjunta y crear la infraestructura administrativa y operativa para la mejora continua, es necesario llevar a cabo un nuevo sistema y concepto de trabajo denominado *Control Total y Mejoramiento de la Calidad*, el cual podemos definir como la búsqueda continua y participativa de la calidad en todos los productos, servicios y trabajos que se planeen desarrollar (ver Fig. 1.4).

La calidad debe aplicarse a todos los niveles de la organización, sin embargo es necesario que sea adoptada una estructura organizacional, la cual ayudará a evitar el desperdicio de esfuerzos y de recursos.

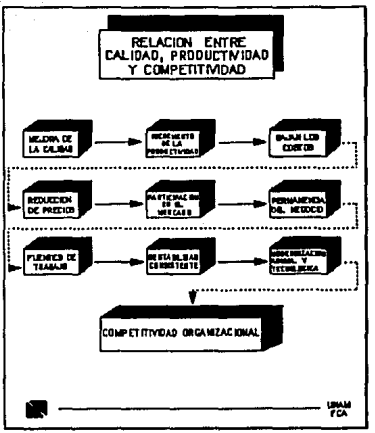


Fig.1.4

1.8 ATRIBUTOS DE CALIDAD DE UN PRODUCTO DE SOFTWARE

Las pautas a seguir para obtener las metas deseadas y las capacidades que debe tener un producto y/o servicio para satisfacer los requisitos del usuario son los atributos de calidad.

A continuación se listan los atributos de calidad de un producto de software:

a) Portabilidad.- Facilidad con la cual un producto de software puede ser transferido de un sistema de cómputo a otro o de una ambiente a otro (independencia de los dispositivos, uso de lenguajes estandarizado).

b) Eficiencia.- Grado con el que el producto efectúa sus funciones mediante un mínimo de recursos de cómputo (óptimo uso de recursos y tiempos de respuesta aceptable).

c) Exactitud y precisión. – Especificación cualitativa de la ausencia de error (correcta selección del lenguaje para la aplicación y correcta selección de tipos de variables numéricas).
Es evidente que la calidad de un sistema tiene una influencia directa sobre la demanda del usuario. Un usuario decepcionado por un mal desarrollo de su sistema generara desconfianza respecto al equipo desarrollador del mismo.

d) Robustez. – Grado con el que un producto de programación puede continuar operando correctamente pese a la introducción de datos no válidos (validación de información de entrada e integración de mecanismos que aumenten consistencia e integridad de la información).

e) Confiabilidad. – Grado en el que un producto esta libre de defectos de diseño y de codificación, o sea, libre de fallas. Grado en el que un producto cumple con los requisitos especificados. Grado en el que un producto cumple con las espectativas del usuario.

f) Amigable. – Que el sistema sea de fácil uso, también que sea autodescriptivo, es decir, que el usuario no requiera de asesoría para poder operar el sistema ya que este cuenta con información suficiente que de una manera sencilla se describa así mismo.

g) Mantenible. – Esta característica se obtiene cuando el código en el que se desarrolla el sistema es entendible, esto significa que dicho lenguaje sea autodocumentado, estructurado, legible y conciso; ademas de que el código debe tener la característica de poder ser modificable en cualquier momento que se requiera.

h) Parametrizado. – Este atributo garantiza que el software desarrollado no requerirá de futuros mantenimientos en relación a los parámetros utilizados en el sistema, ya que al generarlo se estandarizaron estos, con el fin de no tener que cambiar en cada programa y procedimiento las variables cuando estas por aspectos económicos, políticos o sociales varíen, generando en su lugar tablas o archivos a los cuales se les actualizará cuando así se requiera.

Los criterios de aceptación se deben fijar en planes que describen los mecanismos a ocuparse para lograr las metas y los requisitos. Estos criterios deben especificarse de tal manera que sean verificables por métodos conocidos.

Los atributos de calidad de un producto son tan visibles por el usuario como por el personal encargado de su desarrollo. Así por ejemplo, para el usuario encontramos que es muy importante que:

- Satisfaga sus necesidades.
- Sea veloz.
- Tenga ciertos niveles de precisión en los cálculos.
- Sea agradable, estético y amigable en su manejo.
- Se encuentre con un manual accesible y bien documentado.
- Tenga controles integrados.
- Sea parametrizado.

Estos puntos serán satisfechos si la parte cuantitativa interna del producto es bien definida, por ejemplo, para una velocidad adecuada de procesamiento se pueden emplear algoritmos pequeños y/o la compra de coprocesadores.

Ahora bien, desde el punto de vista del personal encargado del desarrollo de sistemas, las cualidades de un sistema pueden ser:

- Código fuente entendible.
- Manuales técnicos (o de sistema) claros.
- Lenguaje de programación adecuado y accesible.

Lo anterior se puede lograr mediante la correcta aplicación de las técnicas de diseño de sistemas.

1.9 FACTORES QUE INFLUYEN EN LA CALIDAD Y EN LA PRODUCTIVIDAD

El desarrollo y mantenimiento de software son tareas muy complejas. El grado de formalidad y la cantidad de tiempo asignada varia de acuerdo al tamaño y complejidad del producto que se va a desarrollar, sin embargo las actividades sistemáticas siempre son necesarias.

La calidad y la productividad del grupo de desarrollo se pueden incrementar al ser considerados los siguientes factores:

- Capacidad individual.
- Comunicación en el grupo.
- Complejidad del producto.
- Empleo de técnicas y notaciones apropiadas.
- Enfoques sistemáticos.
- Control de cambios.
- Nivel tecnológico.
- Nivel de confiabilidad.
- Captación del problema.
- Tiempo disponible.
- Especialización requerida.
- Facilidades y recursos.
- Entrenamiento adecuado.
- Habilidades de los líderes de proyecto.
- Metas apropiadas.
- Administración de proyectos.
- Flujos de mercado.

Todos estos factores influyen en la calidad, el costo y el tiempo de vida del producto.

TESIS

LA CALIDAD EN EL DESARROLLO DE SISTEMAS

El papel del personal en el control de calidad en la fase de desarrollo de sistemas

UNAM
FCA

Con el paso de los años, las empresas vuelcan nuevamente su atención al factor humano, buscando el mejorar el rendimiento de todos los miembros de la organización, trayendo como resultado investigaciones sobre la optimización y rendimiento del personal.

La calidad en el desarrollo de sistemas involucra enormemente al factor humano, ya que es el personal el que se encarga de dar forma a los proyectos de la empresa y por consiguiente, estos deben guiarse por los estandardes de calidad para que sus desarrollos no impliquen altos costos causados por un defectuoso control en el desarrollo incrementando a la empresa desperdicios de hardware, software y personal.

El factor humano es en realidad el corazón tanto del diseño, como del desarrollo y codificación de cualquier sistema y es de vital importancia su participación, ya que en muchos casos se interrelaciona con el mismo sistema.

Los administradores de los proyectos de desarrollo de sistemas, tienen las mismas debilidades humanas, mismas que se reflejan en la calidad del desarrollo.

Cualquier proyecto en el que se involucre la programación, requiere de bases para su administración, estas bases no difieren mucho de las que se aplican a cualquier otro proyecto. Es conveniente que tomemos en cuenta los pasos siguientes, si pretendemos llevar a cabo una buena administración de proyectos.

- Establecer Metas del proyecto.
- Subdivisión del proyecto en subproyectos, así como fijación de los objetivos de cada uno de ellos.
- Establecer un plan de asignación de recursos y funciones.
- Asignación de funciones a cada subproyecto.
- Asignación de recursos a cada subproyecto
- Monitoreo de resultados parciales
- Implantación de acciones correctivas.

Los pasos anteriores del proceso administrativo, incluyendo una retroalimentación se comportan como se presenta en la siguiente gráfica:

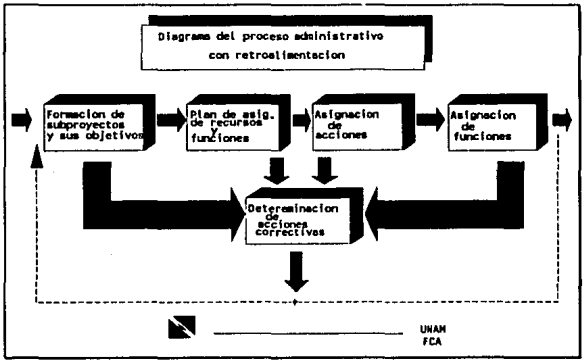


Fig. 2.1

1 Formato de subproyectos sus objetivos y sus metas.
2 Plan de asignación de recursos y funciones.
3 Asignación de recursos.
4 Asignación de funciones.
5 Determinación de acciones correctivas.

Se debe considerar que la duración del proyecto y su desarrollo así como sus recursos, deberán están sujetos a límites de tiempo.

Para el administrador debe quedar bien claro que los beneficios aportados por un proyecto no son palpables sino hasta el final, así que si existen demoras en cualquiera de sus fases críticas establecidas con anterioridad utilizando métodos tales como la ruta crítica y el diagrama de PART, (Program Evaluation and Review Technique) (Técnica de Evaluación y Revisión de programas), se retrasará por obviedad la recuperación de la inversión del mismo, ya que no podemos

perder de vista, que la empresa buscara el saber cuanto le cuesta un proyecto de desarrollo, así como saber cuales serán los beneficios tanto en tiempo como en eficiencia en base a su inversión.

2.1 EL PERSONAL COMO PARTE DE UN SISTEMA

En la actualidad, la complejidad de las computadoras en cuanto a las entradas, como en la interpretación de las salidas y un no menos complejo manejo de difíciles comandos, es por general los problemas a los que se enfrenta un desarrollador de sistemas.

El ser humano, es sin lugar a duda el responsable del producto final de cualquier sistema, así que al termino de un desarrollo ineficiente y de baja calidad traira como resultado que los procesos no den lo esperado por el usuario.

Es conocido en el medio de sistemas que en algunos casos existen estas deficiencias, ya que no son privativas de ninguna empresa, es por esto, que desde el punto de vista del usuario final, lo importante es el obtener un producto que realmente satisfaga sus necesidades, y no como pasa en ocasiones el usuario tiene que conformarse con un producto de baja calidad y por ende que no satisface la totalidad de sus necesidades.
Como ejemplo, en una empresa se requeria de un equipo que pudiera almacenar grandes volúmenes de información para ser consultada por áreas especializadas, esto con el fin de eliminar grandes volúmenes de listados almacenados o de microfichas.

Se opto por desarrollar un sistema de almacenamiento digital, este desarrollo resulto sumamente caro ademas de que no fue evaluado debidamente, trayendo como necesidad para el usuario el generar puestos de operadores de el sistema con una gran dificultad tanto en tiempo como en la calidad de los reportes entregados.

Una forma para incorporar la calidad a un desarrollo se aplica reduciendo la taza de errores, a esto se le conoce como "*Programación defensiva*" la cual maneja como idea básica que ningún programa puede ser escrito sin errores, tomando en cuenta esto, tiene sentido el pensar en que debemos de incorporar la verificación de errores en el software de desarrollo.

Las técnicas de programación defensiva las podemos clasificar en dos ramas importantes:

- Una técnica pasiva, la cual verifica la información en un punto apropiado de un programa.
- Una técnica activa es aquella que busca atraves del programa o de una base de datos periódicamente o durante periodos establecidos en busca de condiciones extrañas.

Existen muchas excusas que dan los programadores para evitar el uso de la técnica de la programación defensiva:

- "Nuestro código tiene pocos o ningún error, asi que la programación defensiva no es necesaria".
- "No es justo incluir verificaciones en mi programa para errores que provienen de fuera, esto es, de errores de programas de otros".
- "La verificación de errores retraza la computadora y requiere de memoria extra".
- "La verificación de errores toma mucho tiempo de programación".

2.2 EXPERIENCIA DEL PROGRAMADOR

El 35 % de los errores que se dan en un producto de programación, son imputables al hardware y software, mientras que solo el 15 % se debe a fallas humanas.

Esto nos obliga a esforzarnos mas en resolver los problemas técnicos que se nos presentan, pero en realidad el factor humano representa la base de un desarrollo exitoso, si tomamos en cuenta que es la claridad del programador

para absorber el requerimiento que se le asigne, observaremos que la experiencia es un factor invaluable para un desarrollo acorde a las necesidades del usuario.

Existen factores importantes que demeritan el desarrollo de un buen programador:

- Limitado desarrollo de la capacitación.
- Insuficiente atención a la motivación y la participación.

La carrera de avance de un programador se basa en la satisfacción de estos factores, ya que mientras mas capacitación y mayor atención a la motivación se tenga, darán pauta a un ambiente adecuado para que un desarrollador ofrezca mayor calidad y entrega de resultados en menor tiempo.

No debemos de pensar que desde un inicio nuestra planta de programadores y desarrolladores debe contar con una gran experiencia, ya que para obtener esta experiencia se requiere de la guia por parte de personal como mayor tiempo dentro el ambiente y una buena capacitación acorde con las necesidades que el puesto requiere.

Es claro el pensar que existen peligros para obtener buenos resultados para nuestra planta de desarrollo, entre otros mencionamos los siguientes:

- Capacitación alejada de las necesidades del desarrollador
- Demora en aplicar lo aprendido.

El primer punto causara que el personal cuente con una serie de conocimientos que no podra aplicar a la problemática que se le presente con los requerimientos de los usuarios, implicando retardos en investigación de las técnicas y herramientas que debera utilizar por cuenta del desarrollador para ofrecer una solución.
De alguna u otra forma el desarrollador tendrá que adquirir los conocimientos necesarios para esto, asi que deberemos tomar muy en cuenta, el diseño de

planes de capacitación acordes a las necesidades que imperen en las instalaciones.

El segundo punto es consecuencia de la falla en el diseño que mencionamos en el punto anterior. Por muy buena que sea una capacitación o entrenamiento asignado a un recurso, estos no dejaran de ser solo conocimientos estancados que en el momento de necesitarse no podran ser aplicados en forma adecuada, ya que el personal habrá olvidado mucho de lo aprendido por la falta de la aplicación de estos conocimientos.

2.3 LOGRO DE LA CALIDAD

Para lograr la calidad lo más económicamente posible, es necesario aceptar y cumplir la responsabilidad a quien corresponda, no en teoria, sino en términos prácticos.

Siempre ha existido la idea de que la calidad es responsabilidad del departamento de control de Calidad o la función responsable de verificar o auditar la calidad o resultado del trabajo. Esto se debe a que el logro de la calidad está basado en un enfoque de detección y corrección del error, en vez de prevención del error.

De quien es responsabilidad la Calidad?
Prácticamente el logro de la calidad es responsabilidad de todos, depende del trabajo de todos. Entonces la única manera de lograr la calidad lo más económicamente posible es que cada departamento asegure su actuación de acuerdo al concepto de calidad. fig 2.2

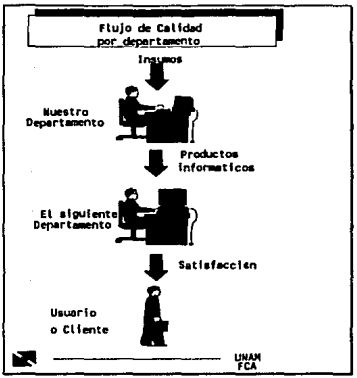


Fig. 2.2

El siguiente proceso es nuestro usuario, esta forma de actuación de cada departamento o proceso, implica utilizar el concepto de calidad en forma interna, osea dentro de la empresa. Por tanto, cada departamento deberá tener claro:

- *¿Quiénes son sus usuarios?*
- *¿Cuales son sus productos, sus especificaciones?*
- *¿Cuál es su proceso y su control?*
- *¿Cuales son sus insumos?*
- *Y manejar indicadores para verificar y asegurar su actuación.*

En otras palabras cada departamento deberá de ejecutar: *CONTROL DE CALIDAD,* para asegurar la calidad de sus productos.

Esta nueva forma de trabajar ha tenido un cambio en el enfoque de control de calidad, y este es el de la "*Detección y corrección de errores*".

Este enfoque significa controlar la calidad al final, o verificar la calidad del resultado del trabajo al final, para separar lo bueno de lo malo. Fig 2.3

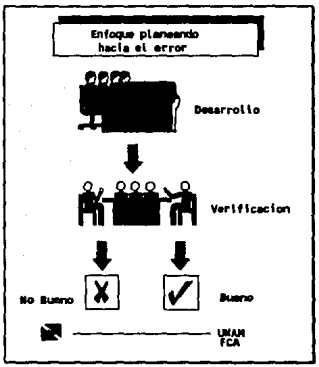


Fig. 2.3

Enfoque: Prevención del error:

Este enfoque significa controlar la calidad durante el proceso, osea durante la realización del trabajo. Lo que implica enfatizar en el control del proceso, para prevenir el error y asi asegurar la calidad del producto informático.

Fig.2.4

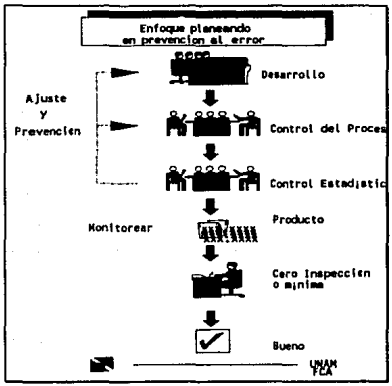


Fig. 2.4

Para controlar la calidad urante el desarrollo, es necesario medir ciertas variables de proceso seleccionadas y caracteristicas de calidad del producto informático previamente definidos para generar datos y monitorear el desarrollo del proceso *(Trabajo)* y asi poder tomar acciones correctivas a tiempo para asegurar la calidad en el resultado *(Producto informático)*.

Este enfoque de prevención de error, que es la forma mas correcta de asegurar la calidad, reconoce simplemente que la calidad se construye en el desarrollo.

2.4 EL CONTROL TOTAL SOBRE LA CALIDAD

Antes de entrar de lleno en materia organizacional, mencionamos que la base para lograr un sistema eficaz para integrar los esfuerzos en materia de desarrollo, mantenimiento y mejoramiento de la calidad realizados por los diversos grupos de la organización, de modo que sea producir bienes y servicios

a los niveles mas económicos y compatibles con la plena satisfacción de los usuarios, esto lo llamamos "*Control de la calidad*".

En el control total de la calidad, se exige la participación de todos los miembros, por lo que en el esquema organizacional propuesto y mas adecuado para un proyecto de software es el de tipo matricial, figura 2.5 esto implica que se tiene que llevar una organización horizontal junto con una organización del tipo vertical.

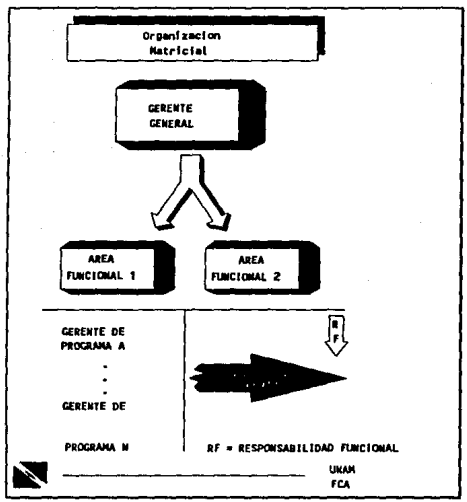


Fig. 2.5

La administración interfuncional es la que suministra un seguimiento que ayuda a fomentar cualquier tipo de relación en la empresa a lo ancho haciendo posible un desarrollo consiente de la garantia de la calidad.

Las caracteristicas principales en la administración de proyectos que no existen en la administración tradicional están representados en la figura 2.6

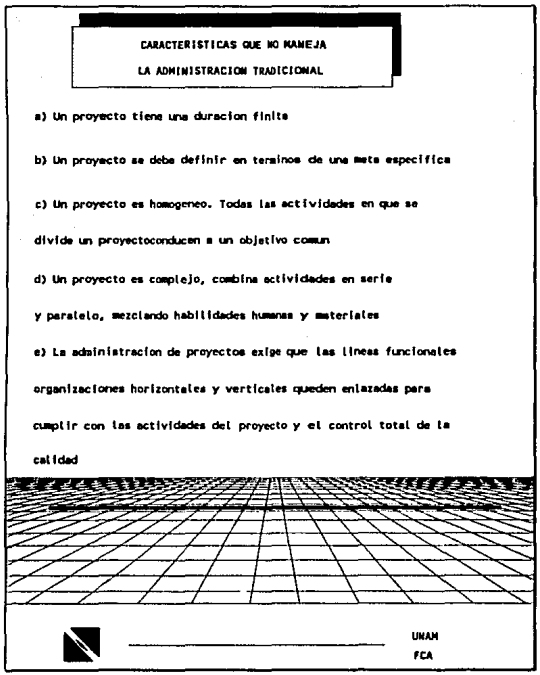


Fig. 2.6

2.5 FORMAS DE ORGANIZACION

2.5.1 GRUPO LIDER PROGRAMADOR

Este tipo de estructura organizacional nació en 1966 por IBM, y la razón para su reacción es la de optimizar la productividad de sus grupos de desarrollo encargados de programación conocida con el nombre de Chief Programmer Teams *(grupo líder programador)*

Esta estructura logra productos con muy alta calidad gracias a una alta productividad que generan sus integrantes. En realidad no todas las estructuras son acordes a todos los proyectos, en si, esta estructura esta pensada por lo menos para el manejo de proyectos de mediano tamaño en adelante.

El líder programador cuenta con su equipo de trabajo o apoyo, y este incluye a los siguientes elementos básicos:

a) Líder Programador. Debe ser una persona conocedora del proyecto, así como una gran experiencia y habilidad. Funge como director técnico del proyecto. El será responsable de generar un reporte de análisis de los requerimientos de la programación del proyecto, asi como de establecer la estructura, del diseño de programación y una codificación
correcta de los módulos mas importantes. Otra de sus características mas importantes es la de hacer una revisión e integración de los productos programados y controlar la calidad y coordinación de esfuerzos de todo el grupo de trabajo.

b) Programador alternativo líder. El se encargara de respaldar al líder programador y en el caso de ser requerido, este se encargara de suplirlo. El estará encargado de hacer la síntesis y un análisis de cuales son las alternativas para la estructura y el diseño de la programación, el apoyara al líder programador en el desarrollo de los módulos críticos. El debera de ser el encargado de elaborar los procedimientos mas adecuados de pruebas [1] ademas de la edición de un manual para el usuario.

c) Bibliotecario. El estará encargado de tener siempre a la mano las facilidades de soporte, así como encargarse de mantener actualizada la biblioteca de programas, así como mantener datos prueba. Se encarga de mantener un registro de los cambios que se efectúen en las bibliotecas durante el desarrollo del proyecto.

d) Especialistas. Estas personas, son personal de apoyo que la empresa contrato de manera externa, ellos cuentan con los conocimientos técnicos específicos sobre la materia del proyecto. La duración de este personal en la empresa se determina por las actividades en donde se involucra sus conocimientos.

e) Programadores. Son los encargados de la codificación de los módulos que no son críticos para el sistema.

2.5.2 ORGANIZACION FORMAL

La estructura que se manejo en el punto pasado, deja de ser funcional cuando la organización se convierte en muy compleja y el proyecto es muy grande. En estos casos lo que requerimos, es de una forma de organización mas elaborada que permitira identificar correctamente las diferentes actividades administrativas y de control de proyectos de las que son propias del desarrollo de proyectos.

Esta estructura esta encabezada por un jefe de proyecto que preferentemente sera un profesional en sistemas que tenga conocimientos tanto de análisis de los requerimientos que se le solicitan, como de los conocimientos técnicos en el área de la computación. Sus funciones son las de controlar y dirigir de acuerdo a una metodología, así como el de apegarse a los presupuestos del área.

Es importante aclarar que esta estructura debera de estar funcionando con la incorporación del concepto básico del grupo líder programador, visto en el punto anterior. Son tres las áreas que se complementan con el grupo del líder programador, integrando una organización formal:

a) Análisis. Estará encabezado por un especialista que domine el análisis del programa a desarrollar. El tendrá como actividad mas importante la de tener la definición de los planes de prueba del sistema. Podemos decir que sus actividades son:

- Definir con los usuario los objetivos del proyecto.
- Elaboración del reporte de análisis.
- Documentar los requerimientos a diseñar.
- Informar al grupo de programación sobre el diseño.
- Definir un plan de pruebas para integrar el sistema.
- Análisis de las pruebas.
- Elaborar el manual del usuario.

b) Desarrollo. Este es un grupo que estará comandado por una persona que este bien capacitada en programación y en general conocimientos de sistemas, ya que este se encargara del diseño de estos bajo técnicas de desarrollo con el fin de producir todos los programas y subprogramas que integraran el sistema final.

Este grupo empleara los planes de prueba generados por el grupo de análisis, y sus actividades especificas serán:

- Definición de la arquitectura a seguir
- Diseñar los programas de computación
- Codificar y documentar los módulos del programa.
- Diseñar una base de datos.
- Elaboración de los procedimientos de prueba.

c) Grupo de control de calidad. El jefe de este grupo, desarrollara las funciones de un auditor que estará al servicio del jefe del proyecto, él se encargara de que se cumplan las normas sean aplicadas y que se este efectuando la documentación

real del proyecto. Cabe señalar que esta auditoria estará siendo aplicada durante el desarrollo del proyecto y no hasta liberar el producto. Las funciones especificas de este grupo son:

- Desarrollo de normas de especificación de requerimientos
- Desarrollo de las normas para el diseño.
- Desarrollo para las normas de la codificación.
- Desarrollo de normas para planes y procedimientos de prueba.
- Formar la biblioteca de soporte de programas.
- Desarrollo y/o adquisición de herramientas automáticas para apoyo en fases de análisis, diseño y desarrollo.
- Control sobre las modificaciones a los requerimientos de programación.
- Realizar auditorias.

2.6 FACTORES QUE LIMITAN LA BUENA APLICACION DE LA CALIDAD

Existen factores que determinan la aplicación correcta de la calidad en el desarrollo.

Un factor importante es el limitado desarrollo de la capacitación a los empleados. En muchas empresas desean que sus empleados sean de los mas eficientes y que trabajen bajo normas de calidad muy altas, pero olvidan que para que esto se de, requieren de garantizar una buena capacitación que de a los empleados los conocimientos y las normas en las que debe basarse éste, y asi de esta forma podra esperar que sus desarrolladores cuenten con todas las herramientas para aplicar una buena calidad.

La atención a la motivación, y la participación son factores que el lider de proyectos debera de tener en mente al pretender un buen equipo de trabajo, mucho personal decae en su productividad al tener una baja motivación a su labor, en realidad, sin una motivación adecuada, el empleado puede convertirse en nuestro peor enemigo, ya que este tratara por muchos medios de bajar su rendimiento, sino es que hasta tratara de sabotear el desarrollo armónico del sistema. Esto involucra una participación mas apegada a la realidad del proyecto, el programador del sistema, no sera el que determine todo un

desarrollo, pero podra aportar su cúmulo de ideas y experiencias que enriqueceran un producto final.

III
Capítulo

TESIS

LA CALIDAD EN EL DESARROLLO DE SISTEMAS

Aplicación de la calidad en el desarrollo de sistemas

UNAM
FCA

3.1 CICLO DE VIDA DEL SISTEMA

El proceso de desarrollo de un producto de software comprende varias consideraciones importantes. La primera de ellas esta constituida por definir un modelo para el ciclo de vida del producto. Este ciclo incluye actividades requeridas para definirlo, desarrollarlo, probarlo, entregarlo, operarlo y mantenerlo.

Cada proyecto en particular, debe tener un modelo de ciclo de vida a seguir, el cual se debe especificar en el Plan del Proyecto, ya que no existe un modelo de ciclo de vida ideal aplicable a cualquier producto.

Una estructura de ciclo de vida de un producto de software como el de la Fig. 3.1 ayuda a una buena elaboración de las politicas de planeación:

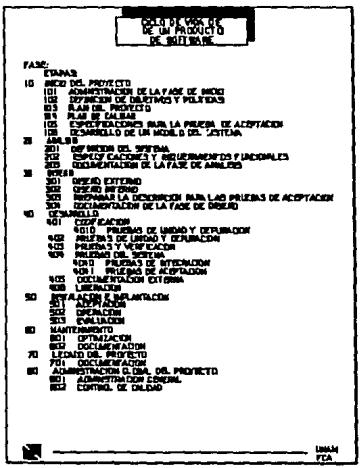


Fig.3.1

Cada fase debe ser delineada y desglosada en etapas de tal manera que tengan un inicio y un final bien definidos y sea posible medir y efectuar controles. Resulta lógico que no todas las etapas se efectúen necesariamente en serie, ya que para optimizar tiempo algunas se pueden realizar en paralelo.

3.2 TECNICAS, HERRAMIENTAS Y TECNOLOGIA DE INGENIERIA DE SOFTWARE

Resolver problemas de una manera organizada y dirigida puede ser difícil porque la gente no siempre está familiarizada con los métodos que paso a paso nos permiten atacar un problema. Cualquiera que sea la actividad intelectual , esta se caracteriza por un conjunto de conceptos fundamentales y de técnicas y herramientas específicas.

Las técnicas son susceptibles de cambio debido a modificaciones tecnológicas, modas intelectuales, condiciones económicas, preocupaciones sociales, etc.
Sin embargo, los principios fundamentales permanecen iguales a través del tiempo, proporcionando las bases para el desarrollo y evaluación de las técnicas.

No es la finalidad de esta tesis tales técnicas y herramientas a detalle, por tal motivo simplemente haremos mención de estas. A continuación presentamos descripciones breves de varias técnicas, herramientas y conceptos fundamentales, a fin de que el lector tenga una idea general de todas ellas.

3.2.1 TECNICAS:

3.2.1.1 A) NOTACIONES RELACIONALES ECUACIONES IMPLICITAS:

(0 < = X < = Y) [ABS (SQRT (X) ^2 .X) < E]

Significa que para todos los valores de X en el rango de 0 a Y, calculando la raíz cuadrada de X y elevándola al cuadrado y restándole X, se obtiene un valor de error permisible.

RELACIONES RECURRENTES:

Consta de una parte inicial (base) y una más o relaciones recursivas. Util para una serie de Fibonacci, formada de la suma de dos números de Fibonacci previos, 0 y 1:

$$F(0) = 0$$
$$F(1) = 1$$
$$F1(N) = F(N-1) + F(N-2) \text{ para todo } N > 1$$

AXIOMAS ALGEBRAICOS:

Los sistemas matemáticos se definen por axiomas. Los cuales especifican las propiedades fundamentales de un sistema y dan la base para derivar propiedades adicionales que estan implicadas por los axiomas, es decir teoremas.

EXPRESIONES REGULARES:
Se pueden emplear para especificar la estructura sintáctica de cadenas de simbolos.
Por ejemplo: *(o(b,c)) denota (ob,ac)*

Utiles para el diseño de compiladores o lenguajes de programación, sistemas operativos y utilerias. Ejemplos de estos son:

- Lenguaje C
- Sistema Operativo UNIX
- Utileria awk de UNIX para analizar textos

3.2.1.2 B) NOTACIONES DE ESTADOS
TABLAS DE DECISION:

Utiles para registrar lógicas de decisión complejas.

TABLAS DE EVENTOS:
Especifican las acciones que se deben tomar cuando ocurren eventos bajo diferentes conjuntos de condiciones.

TABLAS DE TRANSICION:

Para especificar cambios en el estado de un sistema como funcion de fuerzas que los ocasionan. También de gran utilidad durante el trabajo previo en un compilador.

MECANISMOS DE ESTADO FINITO:

El conjunto de diagramas de flujo, expresiones regulares y tablas de transición se pueden combinar para proporcionar un mecanismo de estado finito.

REDES DE PETRI:

Sirven para modelar problemas donde existe paralelismo, concurrencia. Los sistemas concurrentes se diseñan para permitir la ejecución simultánea de procesos, evitando (deadlock) es decir, bloqueo por coincidir en un mismo punto al mismo tiempo.

3.2.2 HERRAMIENTAS

3.2.2.1 HERRAMIENTAS AUTOMATICAS PARA LA ESPECIFICACION DE REQUISITOS:

SREM .- Software Engineering Requirement Methodology
PSL/PSA .- Problem Statement Language/Problem Statement Analizer
RSL/REVS- Requirement Statement Language/Requirement Engineering Validation System
SADT .- Structured Analysis & Design Technique
SSA .- Structured System Analysis

Donde
*SREM*Engloba *RSL & REVS.*

3.2.2.2 DIAGRAMA DE FLUJO:

Un diagrama de flujo es una herramienta utilizada para describir un proceso. Muestra la serie de sucesos que constituyen el proceso, suele comenzar con los insumos, muestra las transformaciones ocurridas a estos insumos y termina con

el producto final. La Fig. 3.2 es un ejemplo de un diagrama de flujo que representa un sistema para dar retroalimentación sobre estudios de calidad de los diseños.

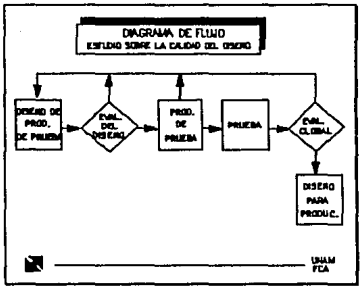


fig. 3.2

3.2.2.3 HOJAS DE VERIFICACION O CHEQUEO:

Son formularios especiales utilizados para reunir datos sobre problemas, tareas terminadas, asignación de trabajo, etc. en la que todos los artículos o factores necesarios son previamente establecidos y en la que los récords de pruebas, resultados de inspección o de operaciones son fácilmente descritos con marcas utilizadas para verificar, por ejemplo; o /. Además de la necesidad de establecer relaciones entre causas y efectos dentro de un proceso de producción, con propósitos de control de calidad y de productividad, las hojas de verificación se usan para:

- Examinar la distribución de un proceso de producción.
- Verificar artículos defectivos.
- Analizar la localización de los defectos.
- Verificar las causas de defectivos.
- Verificación y análisis de operaciones (a esta también se le conoce como lista de verificación).

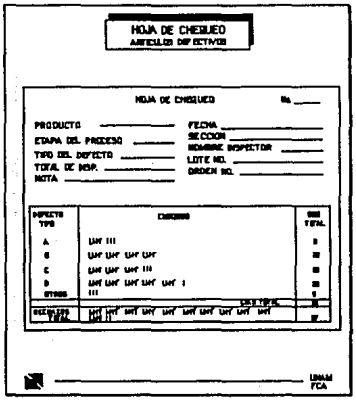


FIG. 3.3

3.2.2.4 ANALISIS DE PARETO:

Diagrama o gráfica que muestra en forma ordenada la ocurrencia de mayor a menor de los factores sujetos a estudio, tales como: fallas, defectos, etcétera. Los diagramas de Pareto pueden aplicarse a todo tipo de mejoras en sistemas o procesos; también sirven para mostrar los resultados de las mejoras, así como, para obtener la cooperación de todos los involucrados; el diagrama de Pareto es una herramienta indispensable para conocer exactamente el objetivo sobre el que debemos concentrar nuestros esfuerzos para el logro de un sistema con calidad.

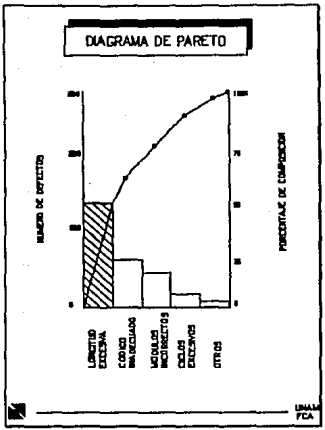


FIG. 3.4

3.2.2.5 SESIONES DE TEMPESTAD DE IDEAS:

Las sesiones de tempestad de ideas son una forma de pensamiento creativo encaminada a que todos los miembros de un grupo aporten ideas sobre determinado tema, y que lo puedan hacer sin ninguna restricción. Las sesiones de tempestad de ideas se rigen por los tres pasos descritos a continuación:

1.- Cada miembro del grupo hace una lista de ideas.
2.- Los miembros del grupo se acomodan en forma circular y se turnan para leer una idea cada vez. El proceso continúa hasta que se hayan leído todos los puntos de todas las listas. Se permite el diálogo únicamente con el fin de aclarar algún punto.
3.- Una vez leídos todos los puntos, el moderador le pregunta a cada persona por turnos, si tiene puntos adicionales que comentar. Este proceso

continúa hasta que a nigún miembro del grupo se le ocurran más ideas.

3.2.2.6 DIAGRAMA DE ESPINA DE PESCADO (DE CAUSA Y EFECTO):

El diagrama de pescado es un medio gráfico para analizar los problemas (efectos) y sus causas (por medio de flechas) que contribuyen a ellos. Este diagrama fué desarrollado por el Dr. Kaoru Ishikawa en la Universidad de Tokio, Japón, en 1953, y desde entonces ha contribuido en la solución de problemas de calidad al mejorar los procesos de producción. El uso de este diagrama facilita en forma notable el entendimiento y comprensión del proceso y, a su vez elimina la dificultad del control de la calidad en él mismo, aún en el caso de relaciones demasiado complicadas y promueve el trabajo en grupo, ya que es necesaria la participación de la gente involucrada para el proceso de desarrollo de sistemas y su uso.

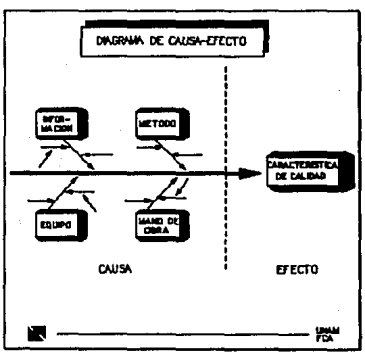


FIG. 3.5

3.2.2.7 HISTOGRAMA:

El histograma representa gráficamente la frecuencia de cada medición dentro de un grupo de mediciones. Es una gráfica de barras que indica una distribución por frecuencia. Dentro de los propósitos para construir un histograma podemos listar los siguientes:

1. Obtener el conocimiento acerca de la distribución de la población(1).en el proceso de producción:
   a. Forma de la distribución.
   b. Localización de la distribución .....media
   c. Dispersión de la distribución .....desviación estándar.
2. Conocer la relación entre los limites de especificación o de tolerancia y la distribución de la población:
   a. Si existe tendencia entre la media de la distribución de la población y el valor medio de los limites de especificación o de tolerancia.
   b. El número o radio de lo defectos.
3. Confirmar los efectos de las mejoras realizadas en el proceso.

(1) El histogramo revela información valiosa respecto al proceso de producción tal como estabilidad de la misma (control), cumplimiento de las normas de operación o procedimientos.

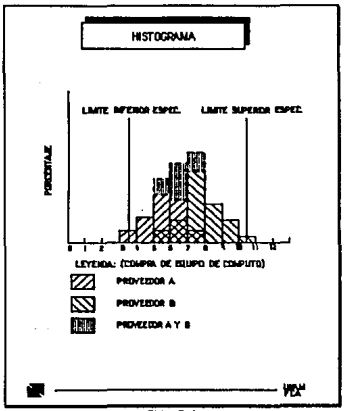


FIG. 3.6

3.2.2.8 DIAGRAMAS DE DISPERSION:

Los diagramas de dispersión muestran la relación o correlación (pero no la causa y el efecto) entre dos caracteristicas cualesquiera. La relación entre dos tipos de datos es fácilmente observable y sus motivos más comunes son analizar:

1. La relación entre una causa y un efecto.
2. La relación entre una causa y otra.
3. La relación entre una causa y otras dos causas.
4. Un efecto y otro efecto.

El diagrama de dispersión tiene mucho uso técnico. Es de gran utilidad para la solución de problemas de calidad en el proceso de desarrollo y en el producto ya terminado, ya que nos sirve para comprobar qué causas (factores) están influyendo o perturbando la dispersión de una caracteristica de calidad o variable del proceso a controlar. También sirve para proceder a su unificación.

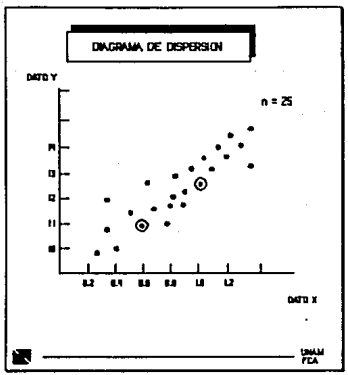


FIG. 3.7

3.2.2.9 GRAFICOS DE CONTROL:

La gráfica de control es una herramienta estadistica que detecta la variabilidad de un proceso. Sirve para solucionar problemas de la calidad en los procesos y para su control. Las gráficas de control son ampliamente utilizadas en la práctica, además de que para su construcción y utilización no se necesita mucho conocimiento de la estadistica. Lo necesario, e importante, es medir bien.

Una gráfica de control consta de limites de control (superior e inferior), establecidos con el propósito de obtener un juicio respecto al comportamiento del proceso; esto es, determinar si es estable o si no lo es, o sea, se está bajo control o fuera de él. Al usar estos limites es posible distinguir desviaciones, tanto por causas asignables, como por no asignables al proceso(2).

Para elaborar una gráfica de control es muy importante distinguir el tipo de datos a graficar. Los datos pueden ser continuos o discretos.

Datos Continuos: Son aquellos que pueden ser representados por cualquier valor dentro de una escala numérica. Ejemplo: mediciones en milímetros, volúmenes en centrímetros cúbicos, pesos de un producto en gramos.
Datos Discretos: Son aquellos que guardan relación con números enteros, basados en conteos. Ejemplo. cantidad de artículos defectivos, número de defectos en un artículo, etcétera.

(2) *W. E. Deming* llama comunes al sistema a las causas no asignables y especiales a las asignables.

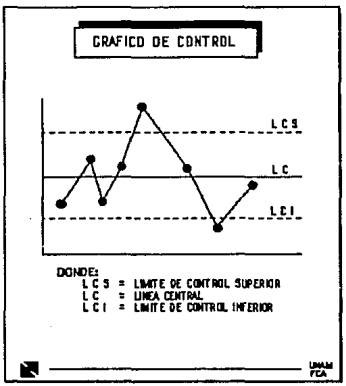


FIG. 3.8

3.2.2.10 ESTRATIFICACION (CATEGORIZACION):

Estratificación es la clasificación de datos tales como defectivos, causas, fenómenos, tipos de defectos (críticos, mayores, menores), en una serie de grupos con características similares con el propósito de comprender mejor la situación y encontrar la causa mayor más fácilmente. Lo importante es entender

que la estratificación es clasificar los datos con el objeto de analizar la causa elegida (en el diagrama de causa efecto) y confirmar su efecto sobre la característica de calidad a mejorar o problema a resolver. La calidad del producto debe ser clasificada en cada una de sus causas. De esta manera, podemos encontrar la relación entre causa y resultado; entonces, podremos definir fácilmente el procedimiento a seguir para mejorar la productividad, la cantidad de producción y la seguridad.

A continuación se muestra el resúmen de algunos conceptos, técnicas, notaciones y herramientas útiles para el diseño de software:

CONCEPTOS FUNDAMENTALES DE DISEÑO

- Abstracción
- Estructura
- Modularidad
- Verificación
- Estética

MODULOS Y CRITERIOS DE MODULACION

- Acoplamiento/Cohesión
- Acoplamiento del contenido
- Acoplamiento de zonas compartidas
- Acoplamiento de control
- Acoplamiento por zonas de datos
- Acoplamiento de datos

NOTACIONES PARA EL DISEÑO

- Diagramas de flujo
- Cartas de estructura
- Diagramas de HIPO
- Patrones de procedimiento
- Pseudocódigo
- Diagramas de flujo estructurado
- Lenguaje natural estructurado
- Tablas de decisión

TECNICAS DE DISEÑO

- Refinamiento por pasos (Nicklaus Wirth)
- Niveles de abstracción
- Diseño estructurado
- Desarrollo integrado TOP-DOWN
- Programación estructurada de Jackson

TECNICAS DE CODIFICACION ESTRUCTURADA

- Una entrada, una salida
- Consideraciones de eficiencia
- Violaciones a una entrada, una salida
- Encapsulado de datos.
- Proposición GOTO
- Recursividad

TECNICAS DE VERIFICACION FORMAL

- Afirmaciones de entrada y salida
- Precondiciones más débiles
- Inducción estructural

ANALISIS ESTATICO

- Métricas del código fuente
- Ecuación de esfuerzo de Halstead
- Métrica ciclomática de McCabe

HERRAMIENTAS AUTOMATIZADAS PARA EL MANTENIMIENTO DE SOFTWARE

- Editores de texto
- Ayudas de depuración
- Generadores de referencia cruzada
- Editores de enlace
- Comparadores
- Sistemas de control de versión
- Configuración de bases de datos administrativas
- Técnicas utilizadas para el control de proyectos
- Ruta crítica

3.3 INFLUENCIA DEL SOFTWARE Y DEL HARDWARE SOBRE LA CALIDAD EN EL DESARROLLO DE SISTEMAS

El término de productividad, utilizado con gran frecuencia últimamente, no significa nada mas el producir algo, sino más que eso, es el hacerlo minimizando recursos(3); sin reducir en ningún momento, la calidad del producto terminado, que en este estudio es el software.

Se debe enfatizar que el término de mínimo y óptimo, aplicados a los recursos, puede crear cierta confusión en el sentido de que el mínimo no es necesariamente el óptimo, por lo que es conveniente establecer cuando se requiere de minimizar y cuando optimizar.

En el caso de algún tipo de contingencia, se requiere contemplar cuál es la configuración mínima del equipo de cómputo con la que la empresa puede mantener sus funciones vitales, esto puede ir en relación con algún convenio interempresarial en caso de desastre, como es el caso del uso del no-break o algún otro sistema de emergencia que entre en operación ante alguna contingencia.

(3). Los recursos que se consideran son de espacio, tiempo, económicos y humanos.

Por otra parte, el mantener un margen óptimo en los recursos puede ser el de utilizar de una manera racional y eliminando, en todo caso, cualquier tipo de desperdicio del equipo de cómputo, evitando dispositivos innecesarios o en su caso subutilización de los mismos.

Respecto a la influencia que ejerce el uso del hardware adecuado sobre la calidad en el desarrollo de un sistema, es de vital importancia determinar el fabricante, el modelo, la capacidad del disco duro, el tamaño de la memoria, la velocidad del equipo, si permite la estructura en red, cuántas terminales o estaciones de trabajo soporta en su configuración, etcétera.

Se debe contemplar la disposición de los dispositivos de la computadora sobre el plano de todas las áreas afectadas, incluyendo la ingeniería de los circuitos de fuerza y aire acondicionado locales y el tendido de los cables que conectan los distintos componentes del sistema.

Referirse específicamente a la productividad del software, es hablar de un programa que, además de cumplir con los requisitos básicos del usuario, lo hace eficientemente, minimizando los recursos utilizados como los accesos a discos para lectura y/o escritura de datos, hasta la cantidad de papel utilizado al emitir un reporte. No se puede reconocer el grado de productividad de todo un sistema, sin haber estudiado minuciosamente cada programa que lo conforma, así como su interconexión entre módulos.

El minimizar u optimizar los recursos provocará una mejor utilización de todo el equipo de cómputo, reduciendo costos en espacio utilizado de memoria, tiempo de ejecución de un programa, papel utilizado para impresiones y algo importante, que es la reducción de costos al momento de darle mantenimiento a determinado programa del sistema para actualizarlo.

La productividad del software se debe tener en cuenta desde el inicio del mismo desarrollo de un programa, por parte de los analistas y de los programadores, en el caso del software desarrrollado. Por el lado del software instalado o adaptado, es conveniente formar un comité de sistemas, integrado por gente de las áreas de Soporte Técnico y Desarrollo de Sistemas, para que de esta manera se tenga una visión más amplia de los conceptos y factores que intervienen en la medición de la productividad del software del sistema. La Fig. 3.9 muestra los efectos del control de software.

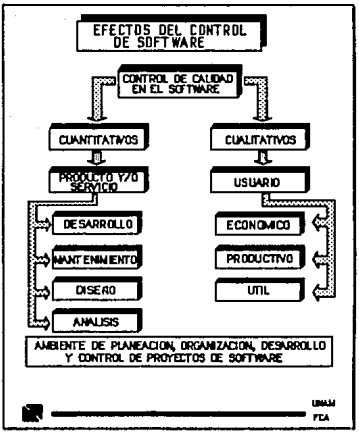


FIG. 3.9

Asimismo, debemos establecer una serie de conceptos de desarrollo y mantenimiento de sistemas que conlleven a implantar una metodologia de trabajo que minimice el tiempo y el costo del mantenimiento y el soporte del sistema desarrollado. A continuación se listan algunos criterios prácticos que pueden contemplarse para disminuir la ocurrencia de errores de los sistemas que seran puestos en producción:

- Depuración de miembros del sistema que no estan en producción.
- Control de versiones de biblioteca.
- Verificación del contenido de bibliotecas con modificaciones.
- Proceso de actualización y registro de modificaciones.
- Estandarización de la estructura del diseño de modificaciones.
- Estandarización de la terminologia en miembros de bibliotecas, procesos, formas, etcétera.
- Depuración de los elementos de proceso redundantes del sistema.

- Elaboración de la documentación detallada de las modificaciones realizadas.
- Desarrollo de textos de ayuda en cada paso nuevo definido o modificación desarrollada.
- Estructuración de las áreas de validación, selección, actualización y reporteo de una manera lógica y eficiente.
- Establecer una lógica de control de puntos de ruptura en el flujo de un proceso. Tanto a nivel de procedimiento como a nivel de programa.
- Establecer un sistema de configuración del sistema en producción a nivel usuario a fin de darle flexibilidad.
- Manejo de los elementos de entrada/salida en forma de bloque de actividades.
- Utilización eficiente de la cola de trabajos.
- Mantener un control preciso de las modificaciones efectuadas a un sistema para eliminar los errores de version.
- Localización de las áreas problema en el flujo de un proceso informálico.
- Establecer politicas de calidad en el desarrollo de sistemas y en el mantenimiento de los mimsos para minimizar y/o facilitar el soporte de aplicaciones en producción.

Puntos a considerar en la optimización a bibliotecas:

- Separar la actualización del reporteo.
- Manejar el control de cifras al principio y final del proceso.
- Puntos de chequeo en mini-procesos.
- Carga de archivos como archivos de grupo.
- Identificar mini-procesos para puntos de reinicio.
- Rediseño de menús con ayuda por cada paso contenido en el menú.

3.4 FUNCIONES FUNDAMENTALES DE LA ADMINISTRACION EN EL DESARROLLO DE SISTEMAS

Siempre que se desee llevar a cabo el desarrollo de un sistema, lo primero que se debe decidir es el objetivo que quiere alcanzarse. Posteriormente se decide

cuál es el trabajo que necesita ejecutarse y cómo será ejecutado, cuáles deben ser los componentes necesarios del trabajo, las contribuciones de cada uno de tales componentes y la forma de lograrlas . En esencia se traza un plan o modelo integrado y predeterminado de las actividades futuras que se realizarán a lo largo del desarrollo del sistema. Todo esto requiere actitud y aptitud para preveer, observar en conjunto, ver intencionadamente hacia adelante. En síntesis es necesaria la *PLANEACION.*

El siguiente paso es distribuir o señalar las actividades necesarias entre los miembros del grupo, para lograr el trabajo. Estas actividades componentes se agrupan y asignan de manera que se realicen con un mínimo de gasto y con el máximo de satisfacción del empleado en su trabajo. Son típicas las preguntas del quién decide qué y cuándo. Este trabajo de señalamiento de tareas, de establecimiento y mantenimiento de relaciones por parte del ejecutivo se conoce como *ORGANIZACION.*

Para llevar a cabo físicamente las actividades resultantes de los pasos de planeación y organización, es necesario que el ejecutivo tome medidas que inicien y continúen las acciones, por el tiempo que sea necesario, para que los miembros del grupo cumplan la tarea. Entre la medidas comúnmente utilizadas por el ejecutivo para poner el grupo en acción, se contarán: el liderazgo, el desarrollo de ejecutivos, la instrucción, la ayuda a los miembros para que se superen por sí mismos y mejoren su trabajo a través de su propia creatividad, y la remuneración. A este trabajo se le denomina *EJECUCION.*

Con el fin de cerciorarse de que se está progresando satisfactoriamente en el trabajo de otros hacia el objetivo predeterminado, los ejecutivos siempre han encontrado conveniente comprobar o seguir estos, ya que pueden presentarse discrepancias, contingencias o malas interpretaciones, para lo cual se deben tomar las acciones correctivas pertinentes Esta función del ejecutivo constituye el *CONTROL.* La acción correctiva podría abarcar uno o todos los aspectos detallados a continuación: cambio de los medios de actuación de uno o más de los miembros del grupo, redistribución de las obligaciones componentes, ajuste de las autoridades delegadadas, alteración del plan del proyecto y modificación de los objetivos previamente establecidos.

3.5 ESTABLECIMIENTO DE POLITICAS PARA EL PROYECTO Y EL CICLO DE VIDA DEL SISTEMA

Una politica es una orientación que fija el campo, los limites y la dirección en la cual se desenvuelve la acción. Sus propiedades fundamentales son:

- Ayudar al logro de los objetivos
- Apoyar el uso del criterio por parte de los subordinados ya que una buena politica debe ser susceptible a ser interpretada.

Establecer politicas para una correcta estructuración de los proyectos, es de gran importancia dentro del esquema administrativo de una empresa dedicada al desarrollo y/o mantenimiento de software. Cabe señalar que un proyecto puede ser:

A. De desarrollo puro: es decir, que debe empezar desde la definición de los requerimientos.
B. De mantenimiento y optimización: implica un ciclo de vida muy similar, la diferencia es que se cuenta con un 90% o más del sistema ya terminado.

3.5.1 POLITICAS DE PLANEACION
A) ESTRUCTURACION DEL PROYECTO

El lider de proyecto y el grupo de control de calidad deben definir un formato con el cual guien los proyectos de desarrollo o mantenimiento de software, para garantizar calidad desde la administración del mismo. Esta estructura debe definir actividades como las siguientes con sus respectivas politicas:

- Determinar el modelo del ciclo de vida del proyecto de acuerdo con el tamaño del proyecto a controlar o a la etapa de desarrollo correspondiente.
- Definir las actividades de administración, control de calidad y validación.
- Conocer la estructura organizacional de su departamento.
- Los requisitos del personal y de los recursos necesarios para el proyecto.

- Programación preliminar del desarrollo utilizando las herramientas que sean necesarias.
- Programación de un estimado de costos para el establecimiento del presupuesto a justificar.
- Los mecanismos de supervisión y control del proyecto.
- Seleccionar del lenguaje o lenguajes de programación a utilizar.
- Señalar los requisitos de pruebas a los que estará sometido el sistema.
- Determinar el nivel de calidad que se desea alcanzar.
- Documentación de apoyo necesaria durante el proyecto y sus etapas.
- Los planes de prueba y aceptación.
- Programación de capacitación al personal o a los distribuidores y los recursos didácticos para la misma.
- El plan de instalación o distribución del sistema o producto.
- Las consideraciones para el mantenimiento de futuras versiones del sistema.

El control de calidad dentro del proyecto requiere del desarrollo de políticas y estándares más específicos para cada fase del ciclo de vida del sistema. Las políticas para cada fase del proyecto son:

a. De definición, especificación de requerimientos y análisis.
b. De diseño.
c. De desarrollo y codificación.
d. De planes de pruebas.
e. De procedimientos de pruebas.
f. De operación y mantenimiento.

B) ANALISIS

Esta es la primer fase del ciclo de vida del proyecto de desarrollo o mantenimiento. Es la fase medular del ciclo y es aquí donde se definen los requisitos y las especificaciones funcionales del sistema. Las actividades que implica la definición del problema son:

1. Desarrollar un enunciado definitivo del problema a resolver, incluyendo una

descripción de la situación actual, restricciones del problema y las metas que se lograrán. Las politicas a seguir en esta actividad son:

- Emplear la terminologia del problema en el enunciado que se defina como definitivo para darle solución.
- Utilizar un recurso humano especialista en el problema a automatizar y un analista de sistemas experimentado, con la finalidad de lograr la interfase de comunicación adecuada, evitando problemas de interpretación y economizando tiempo y dinero. Además de lograr calidad durante la definición del problema.

2. Plantear varias soluciones alternativas para el problema, justificando la solución óptima.
3. Desarrollar una lista de las caracteristicas del sistema o producto a desarrollar, incluyendo sus restricciones.

Aquí resulta de suma importancia que el usuario participe abiertamente en la elaboración de esta lista, ya que estas caracteristicas serán las distintivas del sistema o producto. Es importante mencionar que en estas caracteristicas se deben incluir los atributos de calidad deseables del producto final. Las restricciones dependerán de en gran medida de los recursos que se dispongan y deberán quedar bien definidos para evitar malos entendidos.

Las politicas que se pueden establecer para los puntos 2 y 3 son:

- Documentar cada una de las estrategias de solución del problema, comparándolas y exponiendo los puntos más importantes.
- Dependiendo del tipo de aplicación, se debe desarrollar un prototipo que simule las caracteristicas establecidas para el proyecto.

Presentarlo al usuario, realizar las modificaciones solicitadas y a partir de este prototipo establecer los criterios de aceptación, los cuales serán determinantes para el resto del proyecto.

La definición del problema debe contener el siguiente formato:

1. Definición del problema.

2. Justificación del sistema.
3. Metas del sistema.
4. Restricciones del sistema.
5. Recursos que se proporcionarán (equipo, programación y personal).
6. Características del usuario (nivel del usuario al que se va a dirigir).
7. Ambientes de desarrollo/operación/mantenimiento.
8. Estrategias de solución.
9. Prioridades de las características del sistema.
10. Criterios de aceptación del sistema.
11. Fuentes de información.
12. Glosario de términos.

Una vez que se tiene la definición del sistema, se procede a definir los requisitos para la producción del software, la cual es una especificación técnica de los requisitos que se deben cumplir.

El formato de la especificación de los requisitos para el desarrollo del sistema debe incluir:

1. Panorama del producto a desarrollar y resúmen del mismo.
2. Ambientes de desarrollo/operación/mantenimiento.
3. Interfases externas y flujos de datos que incluyen:
   a. Despliegues y formato de reportes.
   b. Resúmen de comandos del usuario.
   c. Diagramas de bloque o de alto nivel.
   d. Fuentes y destinos lógicos de datos.
   e. Almacenamiento lógico de datos.
   f. Diccionario lógico de datos.
4. Especificaciones funcionales.
5. Requisitos de operación.
6. Condiciones de excepción/ manejo de execpciones /errores.
7. Prioridades de diseño y codificación[4].
   a. Versión prototipo[5].
   b. Versión Uno[6].
   c. Versión óptima[7].
8. Modificaciones y optimización prevista.

9. Criterios de aceptación
   a. Pruebas funcionales y de operación.
10. Estándares de documentación.
11. Guías de diseño (sugerencias y restricciones).
12. Fuentes de información.
13. Glosario de términos.

(4) Dependiendo de la complejidad técnica del problema a automatizar, se puede tener en la fase de diseño y desarrollo versiones sucesivas del mismo sistema.
(5) Resultado de la abstracción de los elementos que forman el problema para presentarlo al usuario y a partir de este prototipo se definen nuevamente algunos criterios operativos y funcionales que posteriormente se integren a la versión final.
(6) Resultado de la abstracción del problema resolviéndolo parcial o totalmente cada uno de los elementos que lo componen. Está sujeto a optimización operativa y funcional en posteriores versiones.
(7) Resultado de la abstracción del problema resolviendo totalmente cada uno de los elementos que lo componen.

C) DISEÑO

Esta es la segunda fase del ciclo de vida del proyecto de desarrollo o mantenimiento. Esta parte es la parte esencial para la codificación del sistema. Esta fase se subdivide en varias etapas que son:

Diseño externo.- En esta fase se concibe planean y especifican las características de un producto de programación. Empieza durante la fase de análisis y continúa en la fase de diseño; es una cambio gradual entre el qué hacer y cómo hacerlo.

Diseño interno.- En esta etapa se incluyen: una especificación de la estructura interna del sistema, detalle de los algoritmos, estructuras de datos, planes de pruebas y guías para la cofificación y mantenimiento.

Diseño arquitectónico- En esta fase se hace énfasis al refinamiento del marco

conceptual del proyecto.

Diseño detallado:- En esta parte del diseño se da gran importancia al detalle de los algoritmos y de las estructuras propias del sistema.

Como el diseño es un proceso creativo que puede ser guiado y dirigido, pero nunca puede reducirse a un procedimiento algoritmico, es decir, las politicas guian, pero no restan creatividad al proceso de diseño.

3.5.2 POLITICAS PARA LAS ACTIVIDADES EN LA FASE DE DISEÑO

1. Revisar la especificación de requisitos. Estudiando las caracteristicas funcionales y los atributos de desempeño deseados en el sistema.
2. Revisar y expander las interfases externas, los diálogos del usuario y los formatos de reportes desarrollados durante el análisis de los requisitos.
3. Revisar y refinar los diagramas de flujo de datos desarrollados durante el análisis de los requisitos y el diseño externo. Identificando los almacenamientos internos de datos y elaborando las funciones de procesamiento.
4. Identificar las abstracciones funcionales y de datos. Registrarlas, usando una notación de diseño.
5. Definir las interfases visibles para cada abstracción funcional y de datos. Registrarlas usando una notación de diseño.
6. Identificar los criterios de modularidad que van a utilizarse para establecer la estructura del sistema.
7. Aplicar las técnicas del método particular de diseño para establecer la estructura del sistema.
8. Iterar los pasos 4 y 7 y cuando sea necesario del 1 al 7 hasta que se logre una estructura adecuada, tratando de cumplir con el tiempo establecido en la planeación del proyecto.
9. Verificar que la estructura del sistema resultante satisfaga los requerimientos.
10. Desarrollar las especificaciones de las interfases para los procedimientos en cada módulo en que se divida el proyecto.
11. Conducir la revisión del diseño preliminar. El nivel de detalle debe ser inclusivo a los niveles I y II de las plantillas de procedimientos, mostrado en la Fig. 3.10

12. Desarrollar representaciones concretas de los datos para los almacenamientos de datos y las abstracciones de los mismos.
13. Expandir las plantillas de procedimientos de la Fig. 3.10, para incluir la información en el nivel III.
14. Especificar los detalles algoritmicos para el cuerpo de cada procedimiento del sistema, utilizando refinamiento sucesivo, diagramas HIPO, pseudocódigo, diagramas de flujo estructurados. Desarrollar representaciones de datos concretas y las interconexiones entre estructuras de datos y algoritmos.
15. Conducir la revisión critica del diseño.
16. Rediseñar cuando sea necesario.

NIVEL I
NOMBRE DEL PROCEDIMIENTO
PARTE DE: (Nombre y número del subsistema)
LLAMADO POR:
PROPOSITO
DISEÑADOR-FECHA(s)
NIVEL II
PARAMETROS: (Nombre, modos, atributos, propósitos)
CONDICIONES DE ENTRADA
CONDICIONES DE SALIDA
GLOBALES: (Nombres, modos, atributos, propósitos)
EFECTOS COLATERALES
NIVEL III
ESTRUCTURAS DE DATOS LOCALES: (Nombres, atributos, propósitos)
EXCEPCIONES: (Condiciones, respuestas)
RESTRICCIONES DE TIEMPO
OTRAS LIMITACIONES
NIVEL IV
CUERPO DEL PROCEDIMIENTO: (Pseudocódigo, diagramas de flujo estructurado y tablas de decisión)

FIG. 3.10

El documento que se genera en la fase de diseño debe contener una estructura como la que se muestra a continuación:

1. Panorama General
2. Referencias
3. Diagrama de estructura
4. Diagrama de flujo de información
5. Descripción del proceso del sistema
   - 5.1 Módulos del programa
     - 5.1.1 Reseña del proceso
     - 5.1.2 Detalle del proceso
     - 5.1.3 Variables externas
     - 5.1.4 Variables globales
     - 5.1.5 Variables locales
     - 5.1.6 Parámetros
     - 5.1.7 Estructura de alto nivel de datos locales
   - 5.2 Total de lineas de pseudocódigo
6. Interfases externas
   - 6.1 Variables externas
   - 6.2 Matriz de variables externas
   - 6.3 Programas externos
   - 6.4 Interfase hombre-máquina
   - 6.5 Mensajes de error
   - 6.6 Condiciones de activación y terminación
   - 6.7 Estructuras de alto nivel de los datos externos
7. Variables y parámetros globales
   - 7.1 Variables globales
   - 7.2 Parámetros globales
   - 7.3 Matriz de variables y parámetros globales
   - 7.4 Estructuras de alto nivel de los datos globales
8. Equipos y lenguajes, sistemas operativos, utilerias, herramientas
9. Recuperación de fallas

10 Certificación de calidad
11 Glosario

D) DESARROLLO

La fase desarrollo la subdividimos a su vez en tres etapas:

- Codificación
  - Documentación interna
  - Pruebas de unidad y depuración
  - Pruebas y verificación
  - Pruebas del sistema
    - Integración
    - Aceptación
- Documentación
  - Externa
  - Interna
- Liberación

Codificación. - Teniendo como objetivo en la etapa de codificación escribir el código fuente y la documentación interna de modo que la concordancia del código con las especificaciones de diseño sean fáciles de verificar, depurar, probar y modificar. Este objetivo puede alcanzarse haciendo el código fuente tan claro y sencillo como sea posible. El sello de los programas de calidad es sencillez, claridad y elegancia, mientras que obscuridad, ingeniosidad y complejidad son indicadores de un diseño inadecuado y un pensamiento mal orientado.

Al utilizar técnicas de codificación estructurada, buen estilo de codificación, documentos de apoyo adecuados, metodología de autodocumentación de programas, librerías, utilerías y procedimientos, además de la correcta selección del lenguaje de programación se logran los tres atributos de calidad mencionados en el párrafo anterior. De otra manera se generará código de baja calidad, pese a lograr los requisitos de diseño y análisis, ocasionando grandes

problemas en fases de mantenimiento y optimización.

El desarrollo de un sistema de alta calidad requiere que el equipo de programación tenga un lider designado, una estructura organizacional bien definida y responsabilidades de cada miembro del equipo de trabajo.

3.5.3 POLITICAS PARA LOGRAR UN BUEN ESTILO DE CODIFICACION

1. Emplear unas cuantas contrucciones estándar de control (*CASE/IF THEN-ELSE/WHILE/REPEAT-UNTIL*). Utilizar máximo tres construcciones para estandarizar el código.
2. Evitar el uso de la instrucción *GOTO*.
3. Nombrar a las variables como entidades explicativas del tipo de valor que contienen.
4. Dar legibilidad al código utilizando estructuras de control bien definidas y cerradas.
5. Aislar la dependencia de la máquina en unas cuántas rutinas System Calls. Para eliminar futuros problemas de portabilidad del sistema a otras máquinas y eliminar exceso de tiempo y mantenimiento.
6. Proporcionar formatos estándar de la documentación para cada subprograma o rutina.
7. Utilizar sangrias, paréntesis, espacios, lineas en blanco y márgenes alrededor de los bloques de comentarios para mejorar la legibilidad.
8. No ser demasiado complicado.
9. Evitar ñas proposiciones *THEN*nulas.
10. No anidar en forma muy profunda.
11. Evitar el exceso de invocación a rutinas con parámetros donde además de la función de la rutina se realicen más tareas que obscurezcan la comprensión de la invocación.
12. No sub-optimizar.

Existen infinidad de rutinas pequeñas dentro de un sistema que tienen un propósito muy especifico dentro del proyecto, que al diseñar correctamente, se reduce en por lo menos un 40% el código, quedando muy poco de este sujeto a optimización. Sin embargo, dada la capacidad de abstracción de cada programador, el tamaño y eficiencia de una rutina de uso general puede ser tan

grande y buena como la capacidad del programador y el tiempo que haya invertido en esta; quedando quizá sujeto a optimización.

DOCUMENTACION

La documentación interna incluye prólogos estándar para unidades de compilación y subprogramas, los aspectos autodocumentados del código y los comentarios internos incrustados en el código fuente. Las notas de cada unidad proporcionan mecanismos para organizar las actividades de trabajo y esfuerzos de documentación de cada programador.

DOCUMENTOS DE APOYO

Las especificaciones de requisitos, documentos, de diseño, planes de pruebas, manuales de usuario, instrucciones de instalación y los reportes de mantenimiento, son ejemplos de documentos de apoyo. Estos documentos son el resultado del desarrollo y mantenimiento sistemático de la programación, las herramientas, técnicas y notaciones para generar y dar mantenimiento a estos documentos, para ello deben definirse y estandarizarse para evitar incompatibilidad y malos entendidos que disminuyen la calidad del sistema. La calidad de los documentos de apoyo es la principal medida para la buena marcha del proyecto.

DOCUMENTACION INTERNA
3.5.4 POLITICAS SOBRE COMENTARIOS

1. Evitar comentarios inmersos en el código al usar:
   - Prólogo
   - Estructuras de programación estructurada
   - Buen estilo de programación
   - Nombres descriptivos del dominio del problema para tipos de datos definidos por el usuario,parámetros formales literales de enumeración,

subprogramas, archivos, etcétera.

2. Escribir comentarios a los bloques de código que:
   * Manipulen datos importantes
   * Simulen instrucciones de control estructuradas que manejen instrucciones *GOTO*
   * Manejen excepciones y errores.
3. Utilizar la terminología del dominio del problema en los comentarios.
4. Emplear líneas en blanco, delimitadores y sangrías para realizar los comentarios.
5. Colocar los comentarios a la extrema derecha para documentar cambios y revisiones, como un estándar que incremente la legibilidad del código durante las revisiones.
6. No manejar comentarios largos y confusos para aclarar un código obscuro y complejo. En su caso, reescribir el código.
7. Asegurarse que el código y los comentarios correspondan uno al otro, así como los requisitos y especificaciones de diseño.

PRUEBAS

Las pruebas de unidad corresponden al conjunto de pruebas efectuadas por un programador de manera individual, antes de la integración de la unidad en un sistema más grande. Existen cuatro categorías de prueba:

a. Funcionales Ejecutan el código con valores nominales de entrada para los que se conocen los resultados esperados, valores límites y valores especiales.

b. Desempeño: Determinan el tiempo de ejecución empleado en varias partes de la unidad, la eficiencia global del programa, el tiempo de respuesta y la utilización de los dispositivos por parte del programa.

c. Tensión: Diseñadas para romper de manera intencionada la unidad.

d. Estructura: Deciden cuáles rutas ejecutar y poder obtener los datos de prueba para llevar a cabo esas rutas.

PRUEBAS DE UNIDAD Y DEPURACION

La depuración es el proceso de aislar y corregir las causas de los errores conocidos. El éxito de la depuración se basa en habilidades altamente desarrolladas en la solución de problemas. Los métodos de depuración son:

a. Depuración por inducción, para lo cual se requiere:

1. Reunir información disponible
2. Buscar patrones que se apeguen en la medidad de lo posible a la realidad
3. Formular una o más hipótesis
4. Demostrar o desechar cada hipótesis formulada
5. Realizar correcciones adecuadas
6. Verificar las correcciones

b. Depuración por deducción.- teniendo como caracteristicas:

1. Listar las posibles causas de la falla
2. Usar la información disponible para eliminar varias hipótesis
3. Elaborar las restantes hipótesis
4. Probar o rechazar cada hipótesis
5. Determinar las correcciones apropiadas
6. Verificar las correcciones

PRUEBAS Y VERIFICACION

El plan de verificación del software contempla lo siguiente:

1. Requisitos que serán verificados
2. Plan de verificación del diseño
3. Plan de pruebas del código fuente
4. Criterios de terminación de pruebas
5. Plan de verificación y documentos
6. Herramientas y técnicas utilizadas en el proyecto

PRUEBAS DEL SISTEMA

Se requiere llevar a cabo una planeación y programación con cierto tiempo de anticipación para asegurar que los módulos podrán integrarse dentro del producto de software desarrollado, cuando assi se requiera.

La estrategia de integración dicta el orden en que los módulos deben estar disponibles, mientras que la estrategia de aceptación implica la planeación y ejecución de pruebas funcionales de desempeño y de tensión para verificar que el sistema desarrollado satisfaga sus requisitos. Estas pruebas de aceptación son realizadas por las organizaciones de control de calidad, los usuarios o ambos. El plan de pruebas de aceptación contempla:

1. Requisitos que se verificarán
2. Casos de prueba para cada requisito
3. Resultado esperado de cada caso de prueba
4. Capacidades demostradas por cada prueba

DOCUMENTACION EXTERNA

Es recomendable que conforme se desarrolla, depura y prueba, se genere la documentación, la cual es de gran importancia y se le debe hacer llegar al usuario final. El formato de esta documentación debe manejar por lo menos lo siguiente:

1. Introducción
   Panorama general del sistema
   Terminología y características básicas
   Resúmen de reportes y pantallas
   Bosquejo del manual
2. Pasos iniciales
   Instalación
   Implantación
   Ayuda
   Ejemplos
3. Modos de operación
   Comandos/diálogos/reportes
4. Características especializadas
5. Sintáxis de los comandos y opciones del sistema

Una vez concluido el desarrollo, documentación y pruebas cumpliendo con las politicas, criterios y estándares establecidos, se libera el sistema o producto de software y, en este momento inicia el ciclo de mantenimiento.

E) INSTALACION E IMPLANTACION

Esta fase se resume en tres etapas:

- Aceptación
- Operación
- Evaluación

Se establece un procedimiento de instalación que cubre los siguientes puntos:

1. Instalación eléctrica regulada para el equipo de cómputo
2. Instalación del sistema operativo
3. Instalación del módulo de ejecución (run time) del lenguaje o medio ambiente propio para el sistema

ORGANIZACION DEL SISTEMA ANTES DE INSTALAR EL PRODUCTO DE SOFTWARE

1. Edición de archivos que contienen parámetros de configuración Kernel o núcleo del sistema operativo.
2. Compilación del Kernel.
3. Creación del perfil de usuarios autorizados y claves de acceso al sistema de cómputo para entrar a operar la aplicación.
4. Creación de los shells o archivos batch para ejecutar el sistema.
5. Dar de alta en la configuración del equipo las terminales o estaciones de trabajo e impresoras.
6. Creación de directorios y subdirectorios para la aplicación.
7. Creación de autorizaciones sobre el directorio y subdirectorios correpondientes.
8. Indicar la información a alimentar inicialmente, a partir de la cual quedará implantado el sistema.

9. Evaluar el sistema, criticando y sugiriendo modificaciones que serán recopiladas para asi ir optimizando el sistema o producto de software desarrollado.

F) MANTENIMIENTO

Fase última y la que completa el ciclo del proyecto. Las actividades del mantenimiento implican mejorar los productos de software, adaptarlos a nuevos ambientes y corregir los problemas presentados, además de mantener un nivel de actualización de toda la documentación generada durante el desarrollo. Esta fase consume cerca del 70% del tiempo y presupuesto del ciclo de vida del proyecto.

Las actividades de desarrollo que propician la calidad en el mantenimiento del software son:

1. Actividades de la fase de análisis.-
   - Fijar objetivos y documentos de apoyo
   - Especificar procedimientos de control de calidad
   - Identificar mejoras del sistema
   - Determinar recursos requeridos para el mantenimiento
   - Estimar costos del mantenimiento

2. Actividades de diseño arquitectónico.-
   - Hacer énfasis en la claridad y modularidad como criterios de diseño
   - Diseñar para propiciar mejoras en el sistema
   - Utilizar notaciones estandarizadas para documentar los flujos de datos, funciones, estructuras e interconexiones
   - Determinar los principios de abstracción de datos, cubrimiento de información y descomposición jerárquica hacia abajo

3. Actividades del diseño detallado.-
   - Hacer uso de notaciones estandarizadas para especificar algoritmos, estructuras de datos y procedimientos para especificar las interfases con otros sistemas

- Especificar el manejo de excepciones
- Proporcionar directorios con referencia cruzada

4. Actividades de implementación o desarrollo.-
   - Utilizar estructuras de una sola entrada y una sola salida
   - Usar sangrias tipo estándar en las estructuras
   - Utilizar un estilo de codificación simple y claro
   - Utilizar constantes simbólicas para asignar parámetros a las rutinas
   - Proporcionar los prólogos estándar de la documentación en cada rutina
   - Apegarse a las guias de comentarios internos

5. Actividades varias.-
   - Desarrollar una guia de mantenimiento
   - Desarrollar un juego de pruebas con su respectiva documentación

Uno de los aspectos más importantes del mantenimiento del software implica rastrear y controlar las actividades del mantenimiento a través de las cuales se puede saber:

- Las versiones existentes de cada producto
- Diferencias entre versiones
- Qué versiones de cuáles productos están distribuidas y en qué instalaciones
- Qué documentos están disponibles para cada versión en cada producto.
- La configuración de hardware que se requiere para operar una versión especifica de un rpoducto
- Causas de las fallas del producto que han sido informadas
- Utilización de bibliotecas para el control de versiones

El mantenimiento exitoso del software, como todas las actividades de la ingenieria del software, requiere de una combinación de las habilidades administrativas y de pericia técnica.

G) RETROALIMENTACION

En la fase de retroalimentación se toman las experiencias vividas durante el

ciclo del sistema y se hace un resúmen de errores y aciertos presentados en todas y cada una de las etapas, siendo éstos documentados para consultarlos en su momento. El resúmen o legado del proyecto debe contener:

- Descripción del proyecto
    - Documentos de salida en cada fase
- Expectativas iniciales
- Situación actual del proyecto
- Identificación de áreas que no han sido atendidas en su totalidad
- Registro de actividades y tiempos estimados
- Lecciones técnicas aprendidas
- Lecciones administrativas aprendidas
- Recomendaciones y sugerencias para proyectos futuros

A partir del legado del proyecto, se va adquiriendo mayor experiencia, con lo cual se van retroalimentando los futuros ciclos, lo cual permite ir puliendo cada vez las políticas, estándares, administración y control de calidad del sistema que se tenga que desarrollar.

3.5.5 POLITICAS DE ORGANIZACION
¿QUE ES UNA ORGANIZACION?

El concepto de organización se puede resumir en lo siguiente:

1. Coordinación de esfuerzos cuyo objetivo es la ayuda mutua.
2. Utilización de los recursos humanos, autoridad, influencia e interrelación entre los miembros del grupo.
3. Integración de las partes de la organización y comunicación.
4. Desarrollo y capacidad de adaptación y de dirección a los cambios.

El problema psicológico de una organización consiste en cómo crear en el personal flexibilidad y adaptabilidad que necesita la organización por desenvolverse en un ambiente variable, ya que el desarrollo de sistemas requiere de una alta capacidad de adaptación a los continuos cambios tecnológicos.
Las politicas de organización resultan importantes debiendo considerar factores tales como:

a. Medio ambiente
b. Formas de organización

A) POLITICAS DE ORGANIZACION APLICABLES A LA FASE DE ANALISIS

1. Se recomienda asignar un analista senior o un grupo de analistas, así como un especialista.
2. Manejar las herramientas necesarias para llevar a cabo un buen análisis:

PSL/PSA (Problem Statement Language/Problem Statement Analizer)
SREM (RSL/REVS) (Software Requirements Engineering Methodology)
SADT (Structured Analysis & Design Technique)
SSA (Structured System Analysis)

3. La calidad podrá ser cuantificada tomando en cuenta:
   * Los criterios establecidos en la planeación de esta fase.
   * Las politicas establecidas en la planeación de esta fase respecto a la

documentación que se debe generar.

Generalmente, en esta fase puede existir cierta holgura de tiempo considerable. Del desarrollo de esta fase depende en buena medida el éxito y la calidad del proyecto.

B) DISEÑO

Establecimiento de politicas para la fase de diseño:

1. Durante esta etapa se recomienda asignar un analista senior o un grupo de analistas, además de un programador senior esto en función del tamaño del proyecto.
2. Definir las técnicas que se van a utilizar para llevar a cabo el diseño del sistema.
3. La medición de la calidad en esta etapa será cuantificada por los siguientes factores:
   * Apego a los criterios establecidos en la fase de análisis, de tal manera que se sigan durante el diseño, evitando errores, retrasos que disminuyen la calidad del sistema.
   * Conocer la estructura organizacional para seguir los flujos de información los canales de comunicación establecidos.

C) DESARROLLO

1. Durante el desarrollo que incluye: codificación, pruebas y documentación, se recomienda asignar a un programador senior y un grupo de programadores, además de integrar personal para pruebas del sistema que se incorpore en el momento de las pruebas, mientras apoya a la elaboración del manual del usuario.
2. Proporcionar las instalaciones, herramientas y equipo necesario para el correcto funcionameinto de la fase.
3. Para medir la calidad de manera cuantitativa en esta etapa se debe:
   * Emplear los recursos tanto materiales como humanos, indicados que contribuyan al correcto desarrollo del sistema

- LLevar a cabo en paralelo la codificación, las pruebas de unidad y la documentación, optimizando con ello las variadas tareas a desarrollar en esta fase.

D) MANTENIMIENTO

3.5.6 POLITICAS PARA EL MANTENIMIENTO DEL SISTEMA

Las politicas para esta fase siguen los esquemas descritos anteriormente en la fasede planeación del sistema. Se cuenta además con implantaciones y personal administrativo adecuado para las tareas generales como la administración del proyecto y recopilación de la documentación de cada fase.

E) RETROALIMENTACION Y LEGADO DEL PROYECTO

POLITICAS PARA LA FASE DE RETROALIMENTACION Y LEGADO DEL PROYECTO

Se elabora un resúmen en donde se describe al lider del proyecto y al circulo de calidad los siguientes aspectos:

1. Tiempos estimados
2. Problemas preentados y posibles soluciones
3. Costos estimados
4. Experiencias y conclusiones
5. Aportaciones y sugerencias

F) CLASIFICACION DE LOS NIVELES DE AUTORIDAD PARA CADA FASE Y CADA MIEMBRO DEL GRUPO DE DESARROLLO

ACT./INT.	LIDER	C.CALIDAD	SUBLIDER	SUBORDINADOS
ANALISIS	1	1	2	2
DISEÑO	1	1	2	3
DESARROLLO	1	1	2	3
MANTENIMIENTO	1	1	2	3

FIG. 3.11

Donde:

1. Autonomia completa
2. Consultar al supervisor
3. Depende de las circunstancias

G) ORGANIGRAMA PROPUESTO PARA LA ORGANIZACION DEL PROYECTO

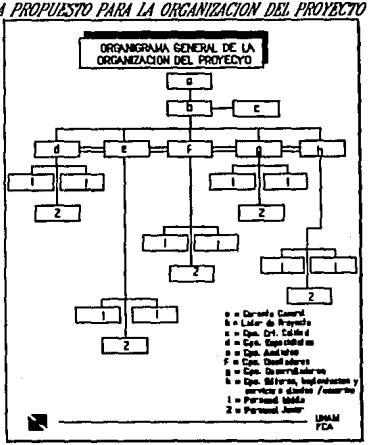


FIG. 3.12

3.5.7 ESTABLECIMIENTO DE POLITICAS SOBRE EL PERSONAL

1. Contar con el personal capacitado para poder hacer frente a los proyectos de análisis y desarrollo de sistemas.
2. Capacitar permanentemente al personal, tomándolo desde un punto de vista de motivación.

3.5.8 ESTABLECIMIENTO DE POLITICAS SOBRE LAS INSTALACIONES', HERRAMIENTAS Y EQUIPO

1. Mantener un ambiente de trabajo armónico, organizado y limpio.
2. Dar mantenimiento regularmente al hardware para evitar contingencias, tales como descomposturas, bajo desempeño y velocidad lenta, que puedan retrasar el tiempo estimado del proyecto e incrementar los costos.
3. Adquirir el software indispensable y evitar la piratería.

3.5.9 ESTABLECIMIENTO DE POLITICAS PARA LA EJECUCION Y DIRECCION

La ejecución es hacer que todos los miembros del grupo contribuyan al alcance de los objetivos y se esfuercen en lograrlos por convicción propia.
Abrahom H. Maslow clasificó a las necesidades en:

a. Fisiológicas
b. Seguridad
c. Afecto
d. Estima
e. Autorealización

El conocimiento de esta pirámide resulta de gran utilidad para el lider del proyecto para tener conocimiento de cúales son las necesidades que satisfacen sus subordinados en su trabajo.

Aspectos importantes que un lider de proyecto debe saber son:

a. La gente difiere en sus necesidades básicas y espera ser tratada como individuo.
b. Es más fácil aceptar cambios e ideas sugeridas si la gente está preparada parallevarlos a efecto.
c. Dos aspectos son muy importantes para explicar el comportamiento de la gente, estos son: los hábitos y las emociones.
d. La gente espera reconocimiento del trabajo desempeñado,cuando así lo merezca.
e. El saberse perteneciente a un grupo aceptable y sentirse importante dentro de el, son factores motivadores para la mayor parte de la gente.
f. El temor es una fuerza motivadora, pero su efecto es negativo y normalmente disminuye con el tiempo.
g. Los empleados mantienen una sensación de bienestar cuando a lo largo de su trabajo realizan logros y pueden demostrar sus habilidades.
h. El personal ejecuta actividades que contribuyen a sentirse orgullo de su trabajo.
i. Fomentar la participación en equipo contribuye al logro de los objetivos de una manera más eficiente.
j. Una persona puede influir sobre el comportamiento de sus compañeros de trabajo.
k. No es recomendable exhibir a los empleados frente a sus compañeros, esta situación crea una sensación de desaliento en el trabajador.
l. Cuando un empleado hace incorrectamente su trabajo, le agrada que le hagan ver su error y que además se le indique la manera correcta de realizarlo.

La dirección se puede definir como la influencia que ejerce una persona sobre otras para que trabajen juntos de manera voluntaria y contribuyan al logro de los objetivos de la empresa en la que trabajan.

Los atributos que debe tener un lider son:

- Inteligencia
- Iniciativa
- Energia o impetu
- Madurez emocional
- Persuación
- Habilidad para comunicarse
- Seguridad en si mismo
- Percepción
- Creatividad
- Participación social

La dirección está formada por tres factores: el lider, los subordinados y la situación.

3.5.10 ESTABLECIMIENTO DE POLITICAS PARA EL CICLO DE VIDA DEL SOFTWARE

1. El lider del proyecto deberá seleccionar al personal que colaborará con él durante el ciclo del sistema. Este grupo de personas deberán estar lo suficientemente capacitados para poder satisfacer los perfiles de los puestos que se requieren en el desarrollo del sistema desde su inicio.
2. El lider del proyecto deberá tener conocimiento de las necesidades de sus subordinados para que en la medida de lo posible pueda satisfacerlas de manera individual.
3. El lider deberá aplicar los motivadores indispensables que propicien la integración y el bienestar de sus empleados.
4. El lider debe fomentar la creatividad y la innovación entre sus empleados.
5. La dirección debe establecer planes de evaluación, compensación y formación de los miembros del grupo, para mantener al personal contento con su trabajo durante el proyecto o mientras éstos permanezcan en la empresa.

3.5.11 POLITICAS EN LA FASE DE CONTROL

Si los resultados obtenidos en el producto final (sistema) que se le ofrece al usuario no van de acuerdo a lo esperado, entonces se deben aplicar medidas correctivas, para que de esta manera se puedan determinar si los esfuerzos administrativos y los procedimientos aplicados en las fases anteriores a la liberación del sistema, están dando los resultados para alcanzar los objetivos del software. Todas estas actividades encaminadas a vigilar, evaluar y corregir en su caso, constituyen la tarea del control administrativo.
El proceso del control tiene como principales funciones, actividades tales como:

- Medición del desempeño.
- Comparación del desempeño con el estándar o politica y de esa manera averiguar la diferencia si es que ésta existe.
- La corrección de desviaciones no favorables a través de acciones que pongan remedio a esas situaciones.

Dentro de los elementos que estan sujetos a medición tenemos:

a. Los resultados de las actividades que componen el proyecto.
b. Los gastos de capital, o sea, la cantidad que se necesita y cual es su utilización efectiva.
c. Los costos, considerando dentro de ellos el presupuesto para el personal y para el equipo de trabajo.

Teniendo como base las politicas, criterios, normas y estándares establecidos durante la planeación y organización del sistema, se comparan los resultados obtenidos contra estos elementos. En el caso de cumplirse, no se aplica ninguna acción correctiva, pero si los resultados obtenidos no son los esperados, entonces tendremos que llevar a cabo actividades que pueden implicar una modificación en la planeación, como podria ser desde redefinir el o los requerimientos hechos por el usuario, objetivo del mismo, alcance del sistema a desarrollar, hasta cambiar algún método en alguna de las fases del desarrollo del sistema.

Otra medida correctiva se podría aplicar sobre la estructura organizacional establecida, ya que la responsabilidad de un individuo tiende a personalizar el trabajo con la autoridad suficiente para tomar las acciones necesarias para alcanzar un desempeño satisfactorio.

La acción remediadora es preferible a la acción correctiva. El proceso de control implica algo más que descubrir las dificultades y corregirlas. En esta fase debe ser descubierta la verdadera causa de la dificultad y hacer todos los esfuerzos necesarios por eliminar las causas de la discrepancia.

De esta manera, se obtiene real ayuda y cooperación; además de lograr una actitud favorable hacia el control.

Para obtener información sobre el desempeño real de las actividades se pueden utilizar los siguientes medios:

- Observación personal
- Reportes verbales (entrevistas, reuniones juntas)
- Reportes escritos

Bajo el concepto de administración por resultados, el empleado participa en el establecimiento de sus propios objetivos y en la determinación de la manera en que pretende alcanzarlos.

Todo esto, se hace con el conocimiento y participación del jefe superior, de tal manera que haya común acuerdo en el establecimiento de los objetivos. Además, los resultados esperados sirven de estándar para evaluar el desempeño.

Este enfoque da un nuevo concepto al control, se convierte en una herramienta para ayudar a guiar los esfuerzos hacia el logro de objetivos, ya que el empleado puede usar esta situación en provecho propio y también en provecho del proyecto.

La administración por resultados incluye la evaluación por resultados. El mejor control es una actividad positiva que el empleado exige con entusiasmo. La mayoría de las personas encuentran satisfacción al hacer un buen trabajo y una de las mejores formas de motivar a un empleado es esperar de él un buen trabajo.

CONTROL POR FACTORES O FUNCIONES

El establecimiento de políticas para llevar a cabo un adecuado control consiste en:

- Determinar si se esta cumpliendo con todos los puntos establecidos en la planeación de actividades de cada fase del desarrollo del sistema.
- Establecer criterios, estàndares y políticas para lograr la calidad en todos los niveles y en todas las actividades
- Planear correctamente el tiempo fijado para las actividades del desarrollo del sistema y comprobar si se està utilizando adecuadamente.
- Cumplir con el costo estimado para el proyecto en la planeación del proyecto.

Un ejemplo del control por factores o funciones se puede apreciar en la Fig. 3.13 que a continuación aparece.

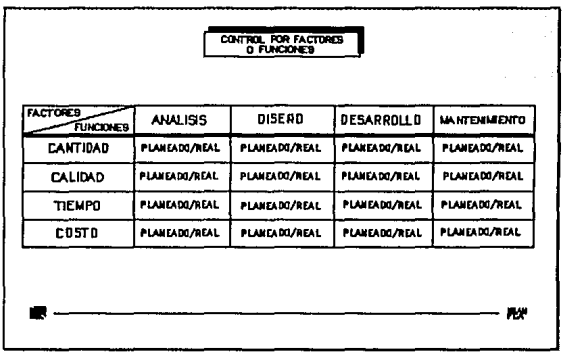


FIG. 3.13

IV
Capítulo

TESIS

LA CALIDAD EN EL DESARROLLO DE SISTEMAS

Efectos económicos del control de calidad en el desarrollo de sistemas

UNAM
FCA

4.1 CONTROL DE COSTOS

En las empresas de nuestros tiempos, el tener un control de costos, es un indicador del buen desempeño de la gerencia. Un error en el que se incurre con frecuencia por parte de la gerencia, es el solo preocuparse de la calidad de su desarrollo, sin recordar que para obtener estos se requiere de un costo, que en muchas ocasiones es muy alto y no tomado en cuenta.

4.2 TIPOS DE COSTOS

Los costos de un proyecto se dividen en tres grandes grupos:

1. Materiales. Son los insumos necesarios que forman parte del producto, o bien, que se requieren mientras se labora. Por ejemplo: Sistema operativo, compilador, manuales, cintas, diskettes, etc.

2. Mano de obra. Constituye el factor humano, por ejemplo: Grupo de desarrollo, Grupo de usuarios, Gerencia, etc.
3. Maquinaria y equipo. Todo el hardware necesario para llevar a cabo el desarrollo

Si tomamos en cuenta estos tres factores, nos daran como resultado un Costo Directo, que ademas esta influido por los siguientes factores:

- Indirectos. Gastos de asesorias, mantenimiento de equipo, personal administrativo (secretaria), papeleria, mantenimiento de las instalaciones, seguridad, etc.

- Otros. Gastos inesperados, horas extras, insentivos a los desarrolladores.

- Utilidad. Se considera un porcentaje que se debe de ganar como un resultado a la inversión de la empresa.

4.3 METODOS DE DISTRIBUCION DE LOS COSTOS GENERALES

A continuación mostramos algunas de las ecuaciones para distribuir costos durante el proyecto:

1. Costo por mano de obra directa:

   COSTO GLOBAL

   HRS. DE MANO DE OBRA DIRECTA

   Expresa en unidades monetarias por hora de mano de obra directa.

2. Costo de mano de obra directa:

   COSTO GLOBAL

   UNIDADES MONETARIAS DE
   MANO DE OBRA DIRECTA

   Expresada como % de gastos generales por costos de mano de obra.

3. Costos de materiales directos:

   COSTO GLOBAL

   COSTO DE MATERIAL DIRECTO

   Expresada como % de los costos por unidades monetarias del material directo.

4. Costo de unidad del producto:

$$\frac{\text{COSTO GLOBAL}}{\text{No. TOTAL DE UNIDADES DEL PRODUCTO}}$$

Expresada como unidades monetarias ($) por unidad del producto.

5. Tasa de máquina:

$$\frac{\text{COSTOS GENERALES DE CADA MAQUINA}}{\text{HORAS DE MAQUINA}}$$

Expresada como unidades monetarias ($) por hora de maquina.

6. Centro de Costos:

$$\frac{\text{COSTOS GENERALES PARA UN GRUPO DE MAQUINAS}}{\text{HORAS DE MAQUINA}}$$

Expresada como unidades monetarias ($) por grupo de horas maquina.

El costo que se genera, es comparado contra el costo estandard respectivo. El costo estándar es calculado por un analista y nos da como resultado los gastos totales del desempeño del trabajo.

Esta es sin duda, la naturaleza de un estándar predeterminado del cual los costos reales pueden expresarse como porcentajes relativos con el COSTO ESTANDAR BASICO como patrón. Para ejemplificar esto, decimos que un costo estándar básico podría ser $ 1,000,000 pesos, pero bajo condiciones de un costo elevado de material y de las tarifas que estén vigentes de mano de obra, el costo real sería de $ 1,500,000 pesos. Conociendo y haciendo una evaluación de la variación, y el costo estándar básico es una base perfectamente válida de control, aunque no representa el costo estándar bajo condiciones prevalecientes normales.

Para corregir las desviaciones de los costos y la reducción de los mismos se logra de la siguiente forma:

- Impidiendo desperdicios de material y tiempo.
- Mejorando los procedimientos y métodos de las opera - ciones.
- Fomentando nuevas ideas para operaciones mas efecti - vas.

4.4 ESTIMACION DE COSTOS EN EL DESARROLLO DE SOFTWARE

Cuando pensamos en el desarrollo de un proyecto de sistemas, debemos tomar en cuenta cosas tales como los recursos que se requieren como son el ámbito donde se desarrollara, el plan de trabajo, el costo, etc.

La tarea de la estimación de costos es una tarea muy difícil y errática en la Ingeniería de Software, ya que intervienen en éste factores desconocidos durante la planeación del mismo, así como una naturaleza intangible del mismo producto.

El usuario siempre nos exigira un presupuesto detallado ya que para él siempre será necesario contar con este dato, a fin de poder realizar su estudio de factibilidad y cubrir los presupuestos para su inversión en cuanto a sistemas.

Ya que se ha realizado las estimaciones, es común que se tenga que aceptar un grado de incertidumbre, no obstante, las estimaciones no han de hacerse de forma empirica ya que existen técnicas útiles que nos permiten tener aproximaciones realistas al costo de nuestro desarrollo.

Para la estimación de los costos en el desarrollo, es indispensable contar con mucha experiencia en el ramo, ademas de tener acceso a la información histórica, entre otras cosas.

Es natural que exista un riesgo en la estimación de los costos, en la fig. 4.1 representamos esta complejidad donde los ejes representan las caracteristicas del proyecto a estimar.

Fig. 4.1

Un factor que realmente dificulta la estimación del costo, es el tamaño del producto, ya que al ser mayor el tamaño del proyecto, crecera la interdependencia entre todos los elementos del desarrollo.

El grado de estructuración del proyecto también tiene efecto en el riesgo de la estimación. Esto se refiere a cual es el grado en que las funciones de nuestro desarrollo se puede compartir y a la naturaleza en cuanto a jerarquía de la información que se debe procesar.

Es de tomarse muy en cuenta lo que se menciona al principio de este capitulo sobre las necesidades de contar con información histórica adecuada, ya que dentro de nuestra estimación podriamos caer en errores cometidos con anterioridad, asi, conociendo nuestras fallas en el pasado, podremos mejorar áreas donde se nos presentaron problemas.

Asi pues, si deseamos hacer una estimación de costos, debemos tomar conciencia de los factores que intervienen, asi como hacerle ver al usuario que cualquier cambio que se realice en el desarrollo nos afectara inevitablemente en el costo final de nuestro producto.

El analista encargado del proyecto se enfrenta ante un gran dilema, ya que para satisfacer los cuestionamientos en cuanto al costo del desarrollo del software, la alta dirección pedira un reporte detallado a éste y dicho análisis toma tiempo y las estimaciones se tiene que entregar de forma inmediata.

Para esto tenemos la Planificación del Proyecto, que no es mas que una serie de herramientas que dan al director un área de trabajo con la podra tener estimaciones con un marco de tiempo determinado. Estas estimaciones se deberan actualizar constantemente en el desarrollo del proyecto.

El alcance del desarrollo debe ser valorado por los analistas del proyecto, ademas que deben de presentar a la dirección de la empresa un amplio detalle de todas los datos cuantitativos como son el número de los participantes, tiempo de terminación, etc, asi como las limitaciones a las que se enfrenta el proyecto

La función y el rendimiento los debemos evaluar al mismo tiempo. Es de considerarse que la función puede tener una gran diferencia para el esfuerzo del desarrollo considerada en el contexto de diferentes limites de rendimiento. (fig. 4.2)
Asi, si tomamos en cuenta que el software interacciona con mas elementos informáticos, debemos tomar en cuenta que el encargado del desarrollo debe considerar la naturaleza y la complejidad de cada interfase con el objeto de determinar los efectos en los recursos y los costos.

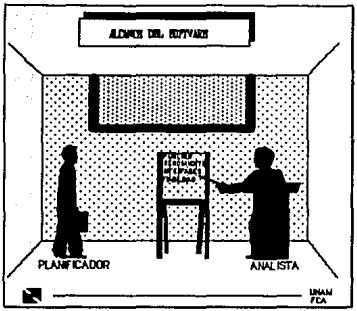


Fig. 4.2

En cuanto a la fiabilidad de cualquier desarrollo de sistemas, se considera muy complejo poderlo determinar ya que siempre estará sujeta a la determinación del hombre, no obstante esto, existen herramientas tales como el tiempo medio entre fallos (TMEF), pero estas no están diseñadas para el estudio del software y se hacen complejas de aplicar.

Si logramos que se efectué una especificación adecuada del sistema, tendremos todos los elementos para poder efectuar una planificación del proyecto de software. En el caso de que esta especificación no se realizara, lo mas importante es que el analista logre efectuar esta especificación.

Otra tarea sumamente difícil para el analista ea la de hacer una estimación aproximada de los recursos con los que se va a contar durante el desarrollo.

Dentro del estudio de los recursos humanos, existe una piramide en donde se sustenta a los elementos como el hardware, como el software como una base para el elemento humano fig 4.3

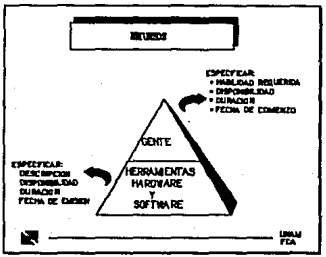


Fig. 4.3

En la actualidad, es sumamente difícil el poder encontrar un número de elementos que puedan desempeñar bien la labor dentro de un proyecto y la gente es el recurso primario del desarrollo del software.

Es de apreciarse que los directivos están presentes en un nivel muy alto de participación al inicio del proyecto y que su participación baja mientras mas elementos técnicos intervienen en este. Es así el caso muy similar con los desarrolladores de mayor experiencia y por último los desarrolladores de menor experiencia están interviniendo mas en los pasos de codificación y de pruebas, esto es muy razonable, ya que los desarrolladores de mayor experiencia deben permitir que los de menor tiempo logren adquirir los suficientes elementos que son dados por la experiencia con el fin de que estos tengan un desarrollo profesional mas integral. En el caso de los licenciados en Informática, debemos de considerar la necesidad de pasar por esta fase que al principio parece tan poco atractiva, pero que al paso del tiempo nos da la capacidad de poder escalar peldaños y tener una mejor colocación en las empresas. fig 4.4

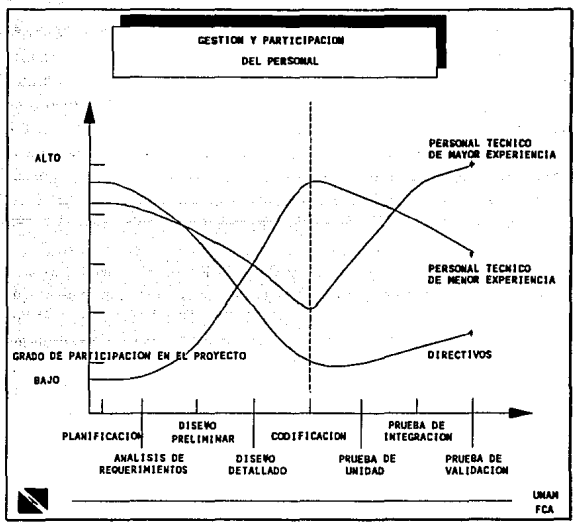


Fig. 4.4

El hardware es la herramienta con la que podemos contar para lograr el funcionamiento de nuestro proyecto de sistemas y por ende para lograr el desarrollo de éste. Para lograr una adecuada selección para nuestro proyecto debemos de

considerar que sistema pensamos desarrollar (tamaño de la aplicación, consumo de recursos de la misma, etc), el equipo con el que contamos, ya que dependiendo de nuestra capacidad y necesidades podremos utilizar un equipo ya instalado o necesitaremos evaluar la necesidad de adquirir uno que cubra esa necesidad.

En las grandes empresas, se acostubra que existan equipos diferentes para desarrollo y producción. En el equipo de desarrollo se tiene un ambiente similar al de produccion aunque con algunas limitaciones, ya que no es necesario contar con los mismos elementos en su totalidad como pueden ser todos los archivos de producción, ya que con muestras se pueden efetuar las pruebas necesarias, e inclusive el equipo de desarrollo puede ser de un tamaño menor en cuanto a la capacidad. En el equipo de producción se cuenta con el ambiente total en donde el sistema puede ser manejado en su totalidad y es con el cual se prestara el servicio deseado.

Las empresas medianas y pequeñas no cuentan con los recursos necesarios para poder adquirir el equipo de respaldo, asi que la forma en la que se tiene que operar, es mediante ambientes de desarrollo dentro del mismo equipo, esto es un poco costo en cuanto a que se tienen dos ambientes similares en el mismo equipo, pero esto mantiene la integridad de los datos de producción. Este ambiente de desarrollo debe estar totalmente asegurado por el responsable de la seguridad del sistema con el fin de que ningun desarrollador pueda tener acceso a areas de producción mientras se esta bajo la etapa previa a la liberación del sistema.

Al igual que el hardware es una herramienta para el desarrollador, tambien el software representa una ayuda de vital importancia para este. En la actualidad se cuenta con sistemas de software diseñado por empresas especializadas en el diseño de paquetes que faciliten las bondades de los equipos de computo, asi como de los lenguajes de programación.

Por lo general, las empresas que diseñan y venden equipos de computo (IBM, NCR, etc) cuentan con equipos de desarrollo

para que los clientes compren junto con el equipo paqueteria que por lo general esta enfocada a mejorar el desarrollo en sus equipos, es el caso de IBM, que para sus equipos macro tiene software facilitador para los programadores, permitiendoles que se preocupen mas en desarrollar, y no en como manejar su equipo de computo.

4.5 ESTIMACION DE COSTOS DEL SOFTWARE DE DESARROLLO

Al comienzo del manejo de las aplicaciones de sistemas, el software representaba un pequeño porcentaje del costo total. Y en verdad asi hera en el pasado, ya que un error fuerte en las estimaciones del costo del software, traia poco impacto para el costo total del proyeto.

La consideración actual debe ser vista muy diferente, ya que el costo actual del software representa tal vez el elemento mas caro dentro de los costos totales del proyecto.

Debemos considerar que cualquier técnica que apliquemos tiene algun grado de riesgo en cuanto a la fidelidad del resultado, ya que ninguna técnica nos podra garantizar la existencia de algun error en la estimación.

La estimación de los costos esta basada en datos historicos guardados en la empresa, la traducción de esta información se convierte en tablas o gráficas en las que manejamos por lo general productividad.

Otra forma de hacer esto es mediante modelos y funciones matematicas que fueron ya probadas por expertos, en este tipo de estimaciones interviene mucho la estadistica como herramienta para eliminar variable innecesarias.

Es tarea del licenciado en informatica el hacer esta dificil tarea de evaluación de costo, no obstante se le considere como de las tareas mas complejas debido a factores como la misma naturaleza intangible del producto final.

El cliente o usuario siempre exigira que le sea presentado con detalle un estudio presupuestal ya que es parte necesaria para el y que este lo pueda incorporar al presupuesto global de la empresa.

4.6 FACTORES DEL COSTO DEL SOFTWARE DE DESARROLLO

Esiste un sinfin de factores que afectan el costo final de un producto de software, la estimación de estos factores es compleja. Entre los factores que influyen de una manera mas directa estan:

1. Capacidad del programador
2. Complejidad del producto
3. Tamaño del programa
4. Tiempo disponible
5. confiabilidad requerida

4.6.1 CAPACIDAD DEL PROGRAMADOR

En proyectos de gran tamaño en donde intervienen una gran cantidad de desarrolladores, las fallas causadas por uno de ellos es compensada por el resto del grupo, pero si tomamos en cuenta un ambiente reducido de personas, cualquier falla sera de gran impacto para el proyecto.

Es competencia de las areas de personal el mantener una planta de empleados con la capacidad requerida para estar al frente de estos proyectos de desarrollo y siempre tener actualizados a estos para minimizar el riesgo de malos elementos dentro del equipo de trabajo.

4.6.2 COMPLEJIDAD DEL PRODUCTO

Al inicio los sistemas no heran diseñados con el apoyo de ninguna técnica de ingenieria de software, asi que los costos para darles mantenimiento hera mucho mas elevado.

No obstante que ahora se utilizan muchas técnicas en las que se logra eliminar mucho la complejidad de los sistemas, los costos que generan ahun son del 40 al 70% dependiendo del sistema. Es esta la razon por la que cualquier medida que podamos aplicar para eliminar la complejidad de un desarrollo tendrá un efecto muy benefico sobre el costo total.

La complejidad de un programa se identifica en terminos de su mantenimiento, el cual se ve afectado por factores tales como:

- El tipo de aplicación, si es nueva o es conocida
- Estabilidad del personal
- Tiempo de vida del software
- Dependencia del programa de su ambiente externo
- Estabiliad del software

Factores técnicos como son:

- Interdependencia de módulos
- Lenguaje de programación
- Estilo de programación
- Tiempo dedicado a la verificación, validación y prueba del programa.

La complejidad del software se mide de acuerdo a las siguientes caracteristicas:

1. Tamaño del programa: Este puno es el mas fácil de medir, se considera que mientras más lineas se tengan de código, mas complejos es un sistema. Aunque en la realidad existen aplicaciones sencillas que requieren muchas lineas de código y que nos son tan complejos, así como programas cortos con una lógica muy compleja.

2. Estructura de datos y flujo de datos: Esta se basa en como están organizados nuestros datos dentro de nuestras aplicaciones.

3. Flujo de control: Es el flujo que nos dan los lenguajes de alto nivel con instrucciones tales como el IF-THEN-ELSE.

Hay diversas clasificaciones de los productos según su complejidad, Brooks establece tres categorias:

A. Programas de aplicación: Incluye procesamiento de datos y programas científicos.

B. Programas de apoyo: Como compiladores, etc.

C. Programas de sistemas: Como son las bases de datos, sistemas operativos, etc.

Según Brooks los de apoyo son 3 veces mas complejos que los de aplicación y estos a su vez son mas complejos que los de apoyo.

Pero esto es realmente ambiguo, por lo que consideraremos a un autor que es mas preciso. Bohem clasifica los programas según el número de instrucciones de código fuente.

El nos da ecuaciones, que en realidad son mas manejables, en ellas maneja constantes y estas son producto de análisis de los datos históricos para muchos proyectos en la

realidad. El tamaño manejado por la muestra era bastante grande, así que las diferencias individuales en la productividad de los programadores es compensada.

Programas de aplicación:
PM = 2.4 * (KDSI) ** 1.05
TDEV = 2.5 * (PM) ** 0.38
Programas de apoyo:
PM = 3.0 * (KDSI) ** 1.12
TDEV = 2.5 * (PM) ** 0.35
Programas de sistema:
PM = 3.6 * (KDSI) ** 1.20
TDEV = 2.5 * (PM) ** 0.32

Donde:

PM = Esfuerzo total en meses de programador requerido en su desarrollo.

KDSI = Número de instrucciones de código fuente entregadas con el producto de desarrollo (en miles)

TDEV = Tiempo para el desarrollo de un programa

CONSTANTES = Las constantes que se presentan fueron materia de un estudio efectuado con varios desarrollos dentro de varias empresas.

Conocido el numero total de meses programador que se requieren para un proyecto y el tiempo nominal de desarrollo requeridos, el nivel promedio que necesitaremos de contratación es obtenido mediante la siguiente división:

NIVEL DE CONTRATACION = PM / TDEV

De manera evidente, debemos adecuarnos a la realidad de nuestro país, ya que estas ecuaciones no son aplicadas con un alto grado de confiabilidad ya que en el medio no se dominan bien las técnicas dadas por la Ingeniería de software. Es labor del Licenciado en Informática el hacer que esta tendencia desaparezca, ya que solo con la consientización por nuestra parte, haremos que esta realidad cambie y que estas herramientas sean tomadas en forma seria, con la conciencia de que su aplicación dará un mejor nivel a los desarrollos en nuestro país. Las nuevas generaciones de lenguajes brindan herramientas que hacen decrecer el tiempo en el desarrollo aportando modelos básicos para fundamentar uno que se deba crear.

4.6.3 TAMAÑO DEL PRODUCTO

Existen ecuaciones similares a las creadas por Bohem. Al observarlas detectaremos que el crecimiento del tiempo de desarrollo es exponencial, osea, no es constante, así tenemos que mientras mas grande sea nuestro proyecto, mas grande será la tasa. fig 4.5.

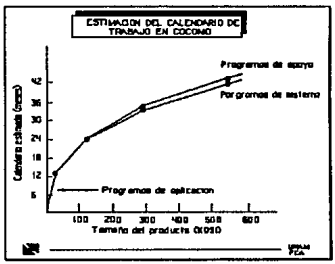


FIG. 4.5

Pero lo importante en este punto es que todos estos métodos serán tan buenos para estimar los costos, como lo sea la capacidad que tenga el licenciado en Informática para hacer una estimación de las instrucciones de código fuente.

4.6.4 TIEMPO DISPONIBLE

Todo el esfuerzo que se le aplica al desarrollo de un proyecto se ve afectado por el tiempo determinado para su conclusión, existen personas que se han dedicado al estudio de un tiempo adecuado para el termino de un proyecto. De estos investigadores la mayoria concluye que sea cual sea el sistema a desarrollar, estos requerinan un mayor esfuerzo si el tiempo de desarrollo se incrementa o se reduce a su valor mas óptimo.

En la fig 4.5 se sugiere una pena muy alta causada por la compresión del tiempo del desarrollo y una ganancia extrema por su expansión convirtiendose en algo ilógico ya que visto así, se predice que no existe ningún esfuerzo para un tiempo no definido de desarrollo.

En la vida real, el modelo de estimación de costos creado por Putnam esta diseñado con técnicas de programación lineal limitando el intervalo de la curva y limitarla dentro de una pequeña región mas restringida alrededor de un tiempo de desarrollo normal y poder jugar con los esfuerzos y el tiempo de desarrollo.
Esta curva es muy sensible, ya que si se incrementa el tiempo de desarrollo mas allá de lo óptimo, en lugar de reducir el esfuerzo total, este crece. Putnam mismo considera que el calendario que se calcula para el desarrollo no se puede comprimirse mas de un 86% del tiempo nominal sin importar la cantidad de gente y de recursos utilizados.

El resto de las curvas de la figura antes mencionada concuerdan mas en lo que se refiere a la compresión del calendario. Pero en realidad, existe un límite en el cual ya no se puede hacer mas pequeño el calendario incrementando el equipo de desarrolladores, este límite es aproximadamente al 75% de comprensibilidad.

4.6.5 NIVEL DE CONFIABILIDAD REQUERIDO

Este permite definir el nivel de confiabilidad de un producto de software como la probabilidad de que un programa desempeñe una función requerida mediante ciertas condiciones especificas y durante cierto tiempo.
La confiabilidad la podemos determinar en términos de exactitud, firmeza, cobertura y consistencia de nuestro código fuente. Las características que se marcan en la confiabilidad se pueden aplicar en un desarrollo de sistemas, debiendo tomar e cuenta que existe un costo asociado al aumento de análisis, diseño, codificación y esfuerzo empleado en la verificación y la validación.

El nivel de confiabilidad se establece en la fase de planeación para considerar el costo de las fallas del programa.

4.6.6 NIVEL TECNOLOGICO

Se considera que el nivel tecnológico es identificable con el lenguaje con el que estemos desarrollando, en México, la mayoría de empresas prestadoras de servicio, manejan lenguajes de no muy alto nivel, pero que son muy prácticos para los desarrollos, entre estos se encuentra el Cobol y el RPG, también influye el equipo con el que se esta trabajando, en nuestro país se comienza a ver que las grandes

corporaciones están iniciando una incorporación de equipos de alta tecnología, debido a que las necesidades del país han cambiado y se requiere adquirir equipos mas acordes con la demanda que los usuarios finales necesita.

La verificación y validación de datos y la documentación generada durante el sistema, ademas de la explotación de las facilidades de los nuevos lenguajes incrementan la confiabilidad, productividad y mantenibilidad, dandonos al final lo que buscamos, la calidad de nuestro desarrollo.

En si tambien el manejo de nuestro equipo facilita a la productividad, ya que mientras los accesos y el manejo de paqueteria amigable sean implementados en nuestras instalaciones, permitiran que nuestro equipo de desarrollo se preocupe por desarrollar y no por aprender a manejar las herramientas con las que estará trabajando.

Bohem menciona que la utilización de equipos y paquetes mas modernos, reducen en un 45% el esfuerzo de nuestro equipo de desarrolladores en comparación con la utilización de técnicas obsoletas.

4.7 ESTIMACION DE COSTOS APLICADOS AL SOFTWARE

Al comienzo de los desarrollos de sistemas, el costo del software representaba un porcentaje muy insignificante en relación del costo total de un sistema, pero en realidad, hoy en dia el software puede ser el elemento mas caro dentro de nuestros proyectos.

Es sumamente complicado el que nosotros podamos considerar una estimación de costos muy exacta en el software ya que en este intervienen factores tales como son politicas de nuestra empresa, técnicas que manejemos y los recursos humanos con los que estamos trabajando.

Hoy en dia podemos cambiar nuestra idea sobre esto, si implantamos pasos sistemáticos que nos ayuden a la estimación adecuada del costo del software, sin perder de vista que tendremos siempre un riesgo aceptable.
Se considera que la mayoria de las técnicas de estimación de costos tuvieron sus orígenes en los años 60's. De alguna u otra forma en todas las técnicas esta presente un factor común que es el que todas están basadas en datos históricos. La forma mas

común de estas es que los datos históricos los conviertan en tablas y gráficas en las que nos muestran la productividad.

Una forma mas de ver estos datos es mediante modelos matemáticos manejados con variables, estos modelos están basados tambien en la experimentación de estos, para reducir mas el numero de variables en los estudios se apoya en procesos estadisticos.

4.7.1 ESTIMACION DEL ESFUERZO

Esta técnica es comunmente utilizada en paises desarrollados, se le considera como la más común para calcular el costo de un proyecto de ingenieria de software. Esta basada en la aplicación de un número de personas-dia, mes o año (dependiendo del analista) a la solución de las tareas individuales del proyecto, a esto se le asocia un costo en pesos a cada unidad de esfuerzo y asi obtenemos un costo estimado.
Consideremos que la tabla fig 4.6 maneja las funciones determinadas por el analista, ademas el debe estimar el esfuerzo de cada una de las tareas y se le aplica un coeficiente de trabajo (costo/unidad de esfuerzo).

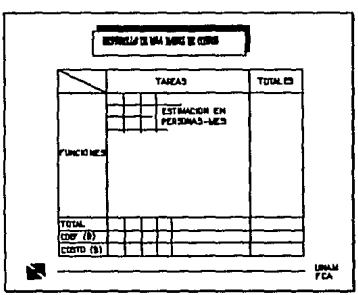


Fig. 4.6

Como ultimo paso se hace un calculo de los costos y el esfuerzo por cada función y/o tarea. Esta técnica preve y sugiere que sus resultados sean comparados con otras técnicas para evaluar la veracidad de la misma.

4.7.2 JUICIO EXPERTO

Esta es una técnica de tipo jerárquica hacia abajo, esta basada en la experiencia, conocimiento anterior así como un sentido comercial de uno o masa individuos de la empresa.

Como desventaja, el experto manejador de esta técnica se puede confiar y pensar que el proyecto en el que esta trabajando, sea similar a uno anterior, olvidando factores importantes y causando que se desvirtúe el sistema al cual se le aplica la técnica, para que esto no pase se sugiere que intervengan varios expertos y así con el juicio de varios, podran trabajar objetivamente.

El trabajar en este tipo de grupos de expertos, causa sin lugar a duda que el grupo se incline por un miembro, para evitar esto, se emplea la técnica DELFI para evitar esta desventaja.

La técnica DELFI fue desarrollada con el fin de manejar el consenso de un grupo de trabajo eliminando los factores que no sean parte de la estimación.

4.7.3 TECNICA WBS

(Work Break Structure) Estructura de división del trabajo, esta es una técnica de tipo jerárquica hacia arriba.

Se maneja como un organigrama jerárquico donde se indican las diferentes partes de un sistema en donde se pueden reflejar una jerarquía de productos o de procesos fig 4.7.

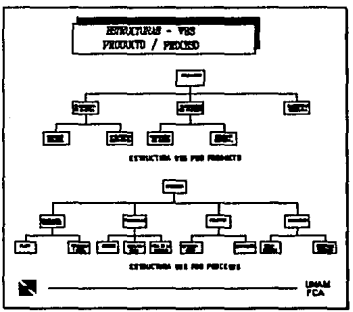


FIG. 4.7

Utilizando esta técnica, la estimación de costos se logra mediante la integración de todos los costos obtenidos en cada componente del organigrama.

La ventaja de la utilización de WBS es el poder identificar los diversos procesos y factores de productos de un sistema, así como la aclaración exacta de que costos se incluyen en la estimación.

4.8 MODELOS DE COSTO BASADOS EN ALGORITMOS

4.8.1 Modelo COCOMO

Este fue desarrollado por Barry Boehm, el nombre de COCOMO quiere decir COnstructive COst MOdel (modelo constructivo de costo) esta técnica se divide en módulos, y estos son:

Modelo 1	Este modelo es un modelo estático, en el se calcula el costo y el esfuerzo del desarrollo de software como función del tamaño del programa en lineas de código estimadas.
Modelo 2	En el se calcula el esfuerzo del desarrollo como función del desarrollo del programa, así como un conjunto de guías en las que se incluyen evaluaciones subjetivas del producto, del hardware, del personal, etc.
Modelo 3	Este se le considera como el modelo avanzado teniendo las caracteristicas del modulo 2 pero ademas incorpora una evaluación al impacto de las guias de costo en todas las fases del proyecto (análisis, diseño, etc).

Ya que esta técnica es muy extensa, daremos una aproximación a esta presentando los primeros dos módulos. Según Bohem, el divide su metodologia para tres tipos de proyectos de software.

- Modo Orgánico: Proyecto pequeño con personal con buena experiencia.
- Modo Semi-acoplado: Proyecto de mediana complejidad y tamaño, en esta deben intervenir personal

de variada experiencia para satisfacer las diferente partes del desarrollo.

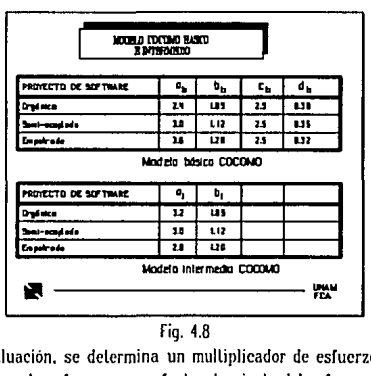
- Modo Empotrado: Esta diseñado para proyectos muy complejos y necesita estar manejado dentro de un contexto muy estricto de software y hardware, así como una serie de restricciones operativas.

Para el modelo 1 del modelo COCOCMO existen ecuaciones que tienen la siguiente forma:

$$E = a_b\ (KLDC)\ \exp(b_b)$$

$$D = c_b\ (E)\ \exp(d_b)$$

Donde E es el esfuerzo aplicado en personas-mes, D es el tiempo de desarrollo en meses y KLDC es el número estimado de lineas de código. Los coeficientes a_b y c_b y los exponentes b_d y d_b están el la fig 4.8.


MODELO COCOMO BASICO
E INTERMEDIO

PROYECTO DE SOFTWARE	a_b	b_b	c_b	d_b
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Modelo básico COCOMO

PROYECTO DE SOFTWARE	a_i	b_i		
Orgánico	3.2	1.05		
Semi-acoplado	3.0	1.12		
Empotrado	2.8	1.20		

Modelo intermedio COCOMO

UNAM FCA

Fig. 4.8

En base a esta evaluación, se determina un multiplicador de esfuerzo y el producto de estos multiplicadores de esfuerzo es un factor de ajuste del esfuerzo (FAE). Los valores para el FAE van de 0.9 a 1.4.

El modelo COCOMO del modelo 2 toma la siguiente forma:

$$E = a_i(KLDC)\ \exp(b_i) * FAE$$

Donde E es el esfuerzo aplicado en personas-mes y KLDC es el número estimado de lineas de código para el proyecto. Los coeficientes están expresados en la fig 4.8.

El modelo constructivo COCOMO sigue una serie de pasos los que están expresados en la fig 4.9.

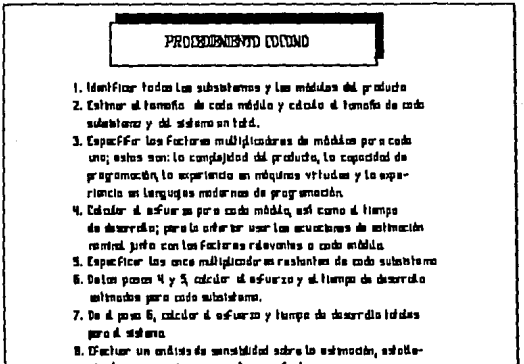


Fig. 4.9

4.9 COSTOS DE NIVEL DE CONTRATACION

En la actualidad es sumamente difícil el poder hacer un cálculo fiel de cuanto personal se requiere para un proyecto, por lo general en cualquier desarrollo de sistemas en la fase de análisis y planeación interviene un grupo muy pequeño de personas, el desarrollo de la arquitectura del sistema intervienen un poco mas de personal y ya para el diseño detallado el grupo se considera bastante grande.

La experiencia con los sistemas es de que ya que se ha terminado la fase de implantación, se requiere de un grupo muy grande de personal que da mantenimiento y este ira decreciendo con el paso del tiempo hasta solo requerir un número muy reducido de personal.

Norden hizo investigaciones sobre el nivel de personal que se requiere para las fases del ciclo de vida de un sistema, mismo que representamos en la fig 4.10.

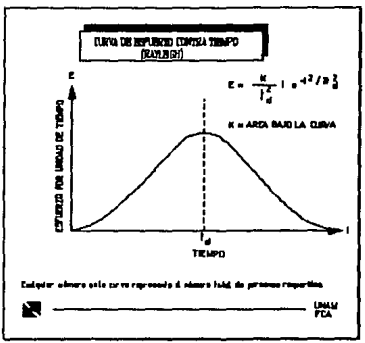


Fig 4.10

TESIS

LA CALIDAD EN EL DESARROLLO DE SISTEMAS

Técnicas de verificación y validación

UNAM
FCA

5.1 INFLUENCIA DE LA VERIFICACION Y LA VALIDACION EN EL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

La función del aseguramiento de la calidad del software tiene la responsabilidad de monitorear los procesos mediante los cuales se desarrollan los sistemas. La función del control de calidad, la cual es parte integral del proceso de desarrollo, compara los productos finales de software con las especificaciones originales y los estándares usados en cada sistema, además informa al grupo desarrollador del software de cualquier defecto que tenga que ser corregido.

El objetivo del aseguramiento de la calidad permite establecer en el proceso de desarrollo de software, la creación de productos (sistemas) libres de defectos.

La Verificación y la Validación (V&V) implican la valoración de los productos de trabajo para determinar el apego a las especificaciones iniciales. Estas especificaciones incluyen los requisitos, la documentación del diseño, principios generales de estilo, estándares del lenguaje de desarrollo, del proyecto y organizacionales, así como espectativas del usuario, al igual que las especificaciones para los formatos y notaciones utilizadas en la especificación de productos diversos.

Se deben examinar los requisitos iniciales del sistema para asegurarse que estos concuerden con las necesidades del usuario, así como con las restricciones del medio ambiente operacional y los estándares de notación. Esto involucra la revisión de todos los documentos de software en cada una de las etapas del ciclo de vida del sistema.

Los objetivos de la verificación y la validación son valorar y mejorar la calidad de los productos de software durante su desarrollo y modificación. Los atributos de la calidad de un producto de software deben ser la corrección, la perfección, la consistencia, la confiabilidad, la utilidad, el apego a los estándares y la eficacia de los costos totales.

La validación es la evaluación del software al final del proceso de desarrollo del sistema para determinar su conformidad con los requisitos.

Hay dos tipos de verificación:

A. La formal y,
B. La del ciclo de vida.

La verificación formal es una rigurosa demostración matemática de la concordancia del código fuente con sus especificaciones.

La verificación del ciclo de vida consiste en determinar el grado en que los productos de trabajo de una fase dada del ciclo de desarrollo cumplen con las especificaciones establecidas durante las fases previas.

La verificación se puede definir como crear correctamente el producto. Así como la validación significa crear el producto correcto.

La alta calidad en un sistema no se puede lograr sólo mediante la prueba del código fuente. Aunque un programa debe estar, en su totalidad libre de errores. En el supuesto de que los errores del código fuente fueran la única medida de la calidad, las pruebas por si solas, no podrían garantizar la ausencia de errores de un programa. La mejor manera de minimizar el número de errores en un programa es encontrarlos y eliminarlos durante el análisis y el diseño, de tal manera que se introduzcan la menor cantidad de errores al código fuente.

Los errores suceden cuando cualquier aspecto de un producto de software es incompleto, inconsistente o incorrecto. Las tres grandes clases de errores de software son los de requisitos, de diseño y de implantación.

La calidad de los productos generados durante el análisis y el diseño, se puede estimar y mejorar utilizando procedimientos sistemáticos (técnicas) de control de calidad, las cuales serán descritas más adelante.

Las técnicas de verificación formal se pueden usar para mostrar de manera rigurosa, que un programa de especificaciones fuente se conforma con sus requisitos; también puede servir para guiar la síntesis sistemática de los programas fuente.

5.2 PLANEACION DE LA VERIFICACION Y LA VALIDACION

Para alcanzar sus máximos beneficios, una organización debe establecer un programa de V&V al inicio de cualquier proyecto de desarrollo de software. Este programa puede iniciarse del mismo modo, durante una fase posterior a cuando el desarrollo está listo. Las siguientes secciones describen los pasos necesarios para planear un adecuado programa de V&V.

5.3 ALCANCE DE LA V&V

Debido a que la V&V cubren una extensa variedad de posibles actividades, el o los administradores del proyecto de sistemas tendrán que encaminar sus esfuerzos a cubrir los propósitos de la V&V en cada proyecto especifico de desarrollo de sistemas. Un programna completo de V&V incluye entre otras actividades las siguientes:

- Verificación y validación de los requerimientos y del diseño.
- Preparación y revisión de los requerimientos, especificaciones y procedimientos involucrados para llevar a cabo la ejecución de las pruebas del sistema.
- Verificación y validación de la documentación como lo son los manuales de usuario, manuales de operador y descripción de programas.
- Verificación y validación del código fuente.
- Integración y aceptación de las pruebas del sistema.
- Revisión de los planes de prueba, procedimientos, especificaciones y reportes.
- Participación en todas las revisiones y auditorias.
- Planeación de las responsabilidades de V&V para el staff de operación y mantenimiento.

La V&V pueden ser aplicadas en cada actividad del ciclo de vida del desarrollo de sistemas. Para ello se requiere de la participación activa y constante del personal asignado para tal efecto durante cada fase. Cuando este compromiso no

fuera posible, como lo puede ser el caso de existir programas posteriores a las actividades de desarrollo, la V&V denominada evaluativa puede ser aplicada. Esta técnica incluye la completa verificación retroactiva, además de la consistencia de la documentación y validación de la correcta operación del software. Las actividades de V&V son constantemente repetidas durante la operación y mantenimiento del sistema, siempre y cuando la requisición de cambios solicitados en el nuevo desarrollo haya sido implementada. Por lo tanto, la V&V comprende el desarrollo completo del ciclo de vida del sistema, todas las actividades de V&V deben ser cuidadosamente planeadas, de tal manera que esten sincronizadas con el proceso tradicional de desarrollo de sistemas. Siempre que un producto de software es producido, el personal de V&V debe estar preparado para revisarlo y ejecutar sus funciones.

5.4 V&V INDEPENDIENTES:

Para proporcionar una evaluación real y honesta de la verificación y validación del sistema , no tendiente a la aprobación completa del software, los administradores del sistema de información deben tener un grupo independiente para llevar a cabo dichas actividades, ya que dificilmente el administrador del proyecto demandará una completa y minuciosa V&V, debido a que esta persona tiene un determinado interés en el desarrollo del software, por lo tanto podría reflejar pobremente los errores presentados en el proyecto.

Básicamente la complejidad y el estado critico del proyecto determinan el grado de independencia requerido. Indistintamente de la independencia elegida, el grupo de V&V debe establecerse de manera anticipada. Asimismo, es importante delimitar el grado de participación que podrá tener el grupo externo que efectuara la V&V.

La administración deberá direccionar el resultado de la V&V independiente durante la fase de inicio del proyecto. Si para llevar a cabo las funciones de V&V no se emplea a un grupo externo, entonces el administrador del desarrollo de sistemas deberá dar a conocer un plan de V&V para asegurar el grado de independencia necesario mismo que se aplicará a la V&V.

5.5 REQUERIMIENTOS DE STAFF

Los miembros del staff administrador del sistema pueden ser seleccionados por el grupo de V&V de la organización en cuestión. El grupo de V&V puede complementarse empleando miembros de staff de la misma empresa o mediante consultores externos. Por ejemplo, el personal operativo de computadoras es cuestionado para evaluar los aspectos operativos de sistemas, los administradores de bases de datos para asegurar que las especificaciones de las bases de datos son correctas, y en muchos otros casos, se consulta con el usuario final para verificar la corrección de los requerimientos. Indistintamente de quiénes estén involucrados en las actividades de V&V, su responsabilidad será la de guiar y asistir al líder del proyecto en la creación de productos de software libres de defectos, a lo largo del ciclo de vida del sistema.

5.6 COMUNICACION/RUTAS DE APROBACION

El administrador de sistemas podrá establecer las rutas de aprobación y comunicación al inicio del proyecto. El administrador de la V&V es responsable de informar al grupo staff o bien al asesor externo de desarrollo, de las deficiencias en el software. La aprobación final y la notificación de las actividades completas de V&V son responsabilidad del administrador del sistema con el soporte del adminsitrador o asesor externo encargado de llevar a cabo la V&V.

5.7 REPORTES DE PROBLEMAS

El grupo de V&V debe preparar las formas para reportar y localizar todos los problemas descubiertos durante el ciclo de desarrollo de sistemas. Dichas formas deben contener la siguiente información:

- Identificación del documento deficiente.
- Localización de la deficiencia en el documento.
- Naturaleza de la deficiencia.

- Nombre de la persona que reporta la deficiencia.
- Fecha del reporte.
- Firma del administrador o consultor externo encargados de la V&V.
- Asignación de la persona responsable de llevar acabo la corrección de la deficiencia.
- Plazo en que deberá cubrirse la deficiencia.
- Prioridad del esfuerzo (opcional).
- Breve descripción del método para realizar la corrección.
- Fecha y firma del administrador o del asesor externo después de realizar la corrección.

Se debe establecer una revisión y aprobación del ciclo del sistema para los reportes de problemas. Cada reporte deberá ser revisado por el personal asigando para la V&V, quiénes darán una muestra de las correcciones hechas en cada deficiencia reportada. Una forma general debe ser usada para todos los reportes de problemas. También debe diseñarse un formato especial para cada aspecto de V&V.

La Fig. 5.1 muestra el ejemplo de un formato usado para el reporte de una discrepancia de software descubierta durante el análisis del código.

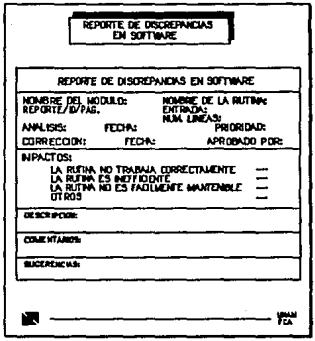


FIG: 5.1

Una copia de este reporte debe ser enviada al área que en la empresa esté encargada de llevar a cabo las funciones de aseguramiento de la calidad, en donde se usará esta información para crear un perfil de defectos por aplicación, de procesos en desarrollo. De esta manera tenemos que el aseguramiento de la calidad permite identificar las partes de los productos (sistemas) que son susceptibles de tener defectos. Consecuentemente, el aseguramiento de la calidad determina el punto en el cual el desarrollo de procesos se está interrumpiendo, así como ver de qué manera se deben reparar esos procesos para prevenir los mismos defectos que pudieran presentarse en futuros proyectos.

5.8 ANALISIS DE REQUERIMIENTOS

Como se ha mencionado, las actividades de V&V deben llevarse a cabo desde los inicios del ciclo de vida del desarrollo de sistemas para poder alcanzar el máximo de beneficios. La primera fase del ciclo de vida que requiere de las actividades de V&V es el análisis de requerimientos. Los métodos para ejecutar este análisis son descritos en las siguientes secciones.

5.8.1 EVALUACION DE REQUERIMIENTOS

Los requerimientos del sistema deben ser definidos en términos de las necesidades del usuario, incluyendo las especificaciones funcionales, requerimientos de ejecución, requerimientos de diseño y análisis del contexto.

Las especificaciones funcionales deben expresar la función a ser ejecutada por el sistema y de qué manera se deben establecer las interfaces con el usuario. Esta situación es aplicable a sistemas dedicados o compartidos indistintamente.

Los requerimientos en el diseño incluyen la configuración del hardware, las especificaciones del sistema para el desarrollo de nuevo hardware y la disponibilidad de los sistemas, así como la capacidad de los datos. Los requerimientos de almacenamiento de memoria principal y secundaria y las alternativas de diseño deben ser provistas en esta evaluación.

El análisis del contexto comprende las interfases externas y el medio ambiente físico que ha sido definido para el propio sistema.

Es muy importante analizar la documentación en la que se dan a conocer los requerimientos del software y verificar que esas especificaciones han sido adecuadamente establecidas para cumplir con la función de la V&V. El análisis de requerimientos es ejecutado tradicionalmente con la ayuda de listas que permiten confrontar actividades y sus elementos , conocidas comúnmente como checklists, las cuales permiten describir los atributos generales de los requerimientos aceptables. Algunos de esos atributos son:

- Consistente.- Cada requerimiento debe contener una notación uniforme, terminología y simbología de acuerdo a los requerimientos del documento que lo respalda.
- Necesario.- Cada uno de los requerimientos debe ser esencial para permitir alcanzar el propósito del sistema.
- Suficiente.- La documentación de los requerimientos debe examinarse para poner cuidado en los requerimientos incompletos o que faltan por cubrirse.
- Factible.- Los requerimientos deben ser analizados para determinar si éstos deben ser implementados usando el hardware y la tecnología con que se

cuenta.

• Probable.- Es decir, el sistema debe permitir crear un ambiente de pruebas para determinar si los requerimientos han sido satisfechos.

• Rastreable.- Esto significa que sea posible de verificar cada requerimiento de software, para saber si éste ha sido generado por los requerimientos del sistema o por las necesidades del usuario.

• Claro.- Cada requerimiento debe ser escrito de tal manera que pueda ser fácilmente entendido.

5.8.2 METODOS ANALITICOS

Los métodos para analizar los requerimientos incluyen comparaciones con la documentación del proyecto en cuestión, sistemas existentes y referencias estándar, modelos analíticos, simulación y análisis oportunos. Los requerimientos del software son casi siempre expresados en lenguaje natural, sólo en algunos casos se emplean herramientas automatizadas para llevar a cabo el análisis de los requerimientos.

5.9 ANALISIS DEL DISEÑO

Cada elemento en el diseño de software debe ser analizado de acuerdo a la capacidad que tenga éste para encontrar las propiedades más convenientes para llevar a cabo el producto final. Esas propiedades y métodos para llevar ese análisis serán discutidos en las siguientes secciones.

5.9.1 EVALUACION DEL DISEÑO

Durante el diseño preliminar, los requerimientos del software son traducidos en el diseño de software del sistema, o en las especificaciones para la ejecución de los programas. La mayoría de los subsistemas son descritos en cuánto a su arquitectura de software, los módulos que los constituyen son definidos en las capacidades funcionales del sistema total, los cuales son colocados entre esos subsistemas y módulos. Algoritmos específicos son usados en cada módulo

seleccionado; estos módulos son colocados de manera oportuna y de acuerdo a los requerimientos. Se establece un alto nivel tanto interno como externo, en relación a las interfases definidas para el sistema, se especifican flujos generales de control y datos a través de las interfases especificadas. Posteriormente el diseño de cada módulo es completado (durante el diseño detallado) en las bases de las especificaciones de ejecución del programa.
Los siguientes elementos forman parte del diseño de software:

- Ecuaciones matemáticas.
- Algoritmos lógicos o matemáticos.
- Descripción de los módulos (entrada/proceso/salida)
- Módulos de interfase.
- Interfases externas del sistema.
- Diagramas de flujo de datos.
- Descripción y referencias de las estructuras de datos.

Estos elementos deben ser analizados usando checklists informales similares a los usados en el análisis de requerimientos.

5.9.2 METODOS ANALITICOS

Varios métodos son utilizados para analizar las propiedades del diseño de sistemas. Los elementos de diseño de referencia cruzada y los requerimientos deben verificar la necesidad y suficiencia de los elementos de diseño. Comparando las partes comunes de los diferentes elementos de diseño debe ayudar para garantizar la consistencia de las interfases. El control lógico debe ser verificado analizando cada trayectoria lógica de los niveles altos del diseño de arriba a abajo (top-down), para que de esta manera se pueda confirmar que las diferentes trayectorias lógicas corresponden a diferentes clases de datos.

Los modelos analíticos que representan matemáticamente al sistema pueden ser usados para evaluar los algoritmos y ecuaciones durantes las etapas de diseño. Estos modelos, ayudan a redefinir los algoritmos especificados en la ejecución de los requerimientos.

Los algoritmos computacionales son verificados por el análisis funcional. Para ello es necesario que se examinen las acciones tomadas por las diferentes clases de datos de entrada de tal manera que éstos deben arrojar los resultados esperados en la salida.

Las ecuaciones matemáticas deben verificarse por medio de la comparación que se hace entre éstas con ecuaciones derivadas y referencias estándar. Este análisis es también útil para ecuaciones que analizan volúmen, distancia y tiempo estimados.

Existen cuatro consideraciones importantes en el desarrollo de un sistema típico, estas son:

1.- Análisis de los requerimientos del sistema.
2.- Análisis del diseño preliminar.
3.- Análisis del diseño del sistema.
4.- Análisis del diseño crítico.

El papel que juega la V&V en el desarrollo de sistemas es, garantizar la adherencia que el desarrollo tiene con los requerimientos del proyecto, así como, verificar que los estándares de la documentación y convenciones sean correctos; identificando, reportando y garantizando para ello, la corrección de las deficiencias que se vayan presentando en su desarrollo.

Es necesario revisar el material para su adecuación a formatos estándar, así como también la claridad de los objetivos del proyecto, el estilo de escritura debe ser aceptable al igual que lo debe ser la consistencia de las referencias cruzadas. El contexto técnico es revisado para incluir en el lo siguiente:

* Planes de desarrollo de software.
* Asignación de tareas y autorización de procedimientos.
* Manejo de planes de configuración.
* Requerimientos y especificaciones.

- Diseño de documentos.
- Especificaciones del diseño de interfases.
- Planes de implementación.
- Estándares en el software.
- Planes y procedimientos de prueba.
- Manuales de usuario.

No necesariamente todos estos puntos se utilizan en la misma revisión, pero si es importante considerarlos en algún momento del ciclo de vida del sistema. Por ejemplo, durante el análisis del diseño preliminar, las descripciones del flujo de un programa de computadora son verificados para llevar a cabo el diseño de las convenciones que son requeridas para el establecimiento de los estándares. La estructura y organización de las bases de datos son detalladas para su correcta identificación y caracterización de todos los tipos de datos, así como para la correcta definición de los tipos y naturaleza de los campos (layouts). Las interfases funcionales, son verificadas para corregir los formatos de mensages y longitud de las palabras. La documentación de prueba es revisada para que ésta sea compatible y adecuada, es decir, que cubra con los objetivos del programa.

Los siguientes documentos podrían ser examinados durante la revisión del diseño crítico:

- Documentación de la descripción de programas (con lenguajes de diseño de programas).
- Almacenamiento de caracteres.
- Documentación para las pruebas.
- Requerimientos para las pruebas.
- Especificación detallada del diseño de interfases.

El staff de V&V verifica el criterio que se tiene para el análisis del diseño específico. El chequeo debe ser hecho en forma manual a través del análisis que se haga a toda la documentación del diseño, incluyendo los planes para el desarrollo del software y adecuando éstos a los estándares y convenciones establecidas para el sistema a desarrollar. El uso de lenguajes para la

especificación de los requerimientos y el diseño de programas facilita la verificación de los procesos y permite el uso de herramientas automatizadas para el análisis.

El análisis del diseño debe revelar deficiencias en el mismo diseño, el cual está relacionado con la documentación. Algunas de esas deficiencias deben indicar incumplimiento en los formatos y convenciones estándar, los cuales deben ser fácilmente corregibles. Otras deficiencias deben violar la modularidad o la jerarquía del sistema, los estándares del diseño top-down, así como requerir costos adicionales por rediseño. Muchos de los más serios problemas pueden prevenirse, implementando una revisión activa del análisis durante la fase de diseño del desarrollo de sistemas, para confirmar que el criterio del diseño esta siendo satisfecho.

5.10 ANALISIS DEL CODIGO

Durante la etapa de codificación, el diseño detallado es traducido en código fuente y toda la compilación o errores de ensamblamiento son eliminados.

Generalmente, el programador aisla la prueba de unidad de cada componente de un programa de la ejecución, de esta manera se deben remover todos los errores obvios antes de integrar el resto del programa. Una de las funciones de la V&V en la prueba de unidad del código fuente, es la identificación de los errores criticos. El grupo encargado de la V&V no debe esperar hasta que las pruebas de unidad de los programas es completada para llevar a cabo el estudio y planeación de la V&V.

5.10.1 VERIFICACION

La verificación del código es un proceso manual en el cual se lleva a cabo una revisión de los listados fuente, estableciendo su relación con el diseño detallado y con cada sección que comprende los requerimientos del proyecto.

El análisis del código examina tanto el lenguaje del programa fuente como la

compilación o código objeto. Las ecuaciones de los programas y la lógica son reconstruidas manualmente o usando ayudas automatizadas y son comparadas con las especificaciones de diseño; ese proceso identifica los errores generados al trasladar el diseño en lenguaje de programación.

El análisis del código permite descubrir errores en la programación, tales como el uso de una instrucción por otra, programación errónea o violación a los estándares de la programación.

Cada listado de programa debe ser analizado en contenido y estilo. El análisis de las técnicas de corrección de programas es una de las funciones de la V&V. La revisión inicial de el listado fuente y de los estándares y convenciones de codificacòn debe llevarse a cabo tan pronto como sea posible, una vez que cada componente es introducido en la biblioteca que contiene a los programas a revisar.

Debido a que los componentes de los programas cambian continuamente, los listados de los programas deben ser revisados periódicamente para evitar violaciones posteriores a la revisión inicial. Para asegurar la calidad de un sistema, es indispensable revisar siempre que haya oportunidad, los listados de todos los programas fuente antes de llevar a cabo la configuración física.

Existen métodos manuales o técnicas automatizadas que pueden ser usadas para la revisión del código. La elección depende del número y tamaño de los listados a ser revisados, así como de la disponibilidad de las herramientas de automatización para revisar el código escrito en un lenguaje particular de programación.

La persona encargada de llevar a efecto la revisión del código deberá poner especial cuidado en la búsqueda de problemas tales como los que a continuación se listan:

- Uso de lenguajes de programación no autorizados
- Uso de código no estructurado, donde no debe permitirse
- Violación en los nombres de las convenciones, por ejemplo en las bases de datos, programas, símbolos o

enunciados
- Asignación incorrecta de columnas
- Violación en las convenciones para el uso de espacios en blanco o ceros
- Inicialización incompleta o incorrecta de la memoria
- Uso de técnicas de entrada/salida no autorizadas
- Comentarios insuficientes en los programas

5.10.2 VALIDACION

La validación comprende la práctica y el análisis del software, para garantizar su ejecución de acuerdo a las especificaciones y para permitir el descubrimiento de los errores latentes. Los dos tipos de validación son:

a. Análisis Estático y,
b. Análisis Dinámico

A) ANALISIS ESTATICO.-

El análisis estático es una técnica para valorar las características estructurales del código fuente, o cualquier representación notacional que se apegue a las reglas sintácticas bien definidas. Incluye técnicas que proporcionan información general acerca de un programa (tablas de referencia cruzada) o búsqueda de errores particulares (variables no inicializadas).

Existen tres clases de análisis estático, mismos que serán descritos a continuación.

1. Análisis de la Estructura Lógica

El análisis de la estructura lógica es usado para aclarar la estructura lógica interna de un programa. Sin embargo, esto no identifica los errores específicos en el código. Este tipo de análisis incluye las herramientas tradicionales de programación como los son: las tablas generadoras de símbolos, analizadores de la dependencia de los subprogramas, generación de referencias externas con

generadores de tablas de referencias cruzadas y rutinas automáticas de diagramas de flujo.

2. Análisis Estático de Errores
Este método es usado para identificar errores graves en un programa (variables inicializadas, violación en la especificación de estándares y convenciones de programación o código no ejecutable). Las herramientas automatizadas pueden usarse para encontrar una clase en particular de error. Los estándares de programación hacen incapié en la verificación de los estándares y convenciones de la especificación de programas. Los analizadores de la estructura de los programas ejecutan rutinas, mediante las que se pueden identificar los loops de programación, asi como el código que puede ser innecesario o no ejecutable en un programa.

Las trayectorias lógicas individuales en un componente de programa deben identificarse a través del uso de analizadores de trayectorias. La mayoria de los compiladores que ejecutan el chequeo de errores substanciales y diagnósticos, son también usados por el análisis estático de errores.

3. Ejecución Simbólica.-
Esta técnica es usada para validar la exactitud de la lógica y de los algoritmos computacionales. Las variables numéricas son permitidas para tomar valores simbólicos y numéricos. Un valor simbólico es una expresión algebraica o una cadena de caracteres (texto) usada por el programador para mantener el valor de una variable.

Cuando un programador ejecuta simbólicamente un programa, el programador elige las trayectorias para analizar y asignar los valores de las variables de entrada. La ejecución simbólica de una trayectoria se lleva a cabo mediante una evaluación simbólica de la secuencia de la asignación de las instrucciones presentadas en una treayectoria. Una expresión aritmética o lógica es evaluada de manera simbólica por medio de la sustitución que se hace de los valores simbólicos de las variables en una expresión por las variables actuales.

B) ANALISIS DINAMICO

El análisis dinámico involucra la instrumentación del código fuente, de tal manera que los programas proveen de la información interna referente a su funcionamiento mientras se estan ejecutando. Las herramientas de el análisis estático pueden usarse con la mayoria de los compiladores.
Un programa modificado debe contener tanto el código fuente original como el que se le inserta para su corrección y posterior compilación, ensamblamiento y ejecución. El tiempo de ejecución estático puede ser recuperado al llegar la terminación del programa o mientras el usuario mantenga el control del mismo.

Hasta aquí, el analizador genera un perfil para la ejecución de un programa, a través de la alteración del código fuente para añadir rasgos a la programación (contadores, salidas selectivas, etcétera.) para determinar por ejemplo, qué enunciados o propocisiones y subrutinas en el código han sido ejecutadas y con qué periodicidad, también permite conocer el rango que asumen los valores de las variables. Este perfil permite localizar los errores y ayuda a dimensionar la eficiencia de las pruebas.

5.11 PRUEBAS
EL ROL DE LA V&V EN LAS PRUEBAS DEL SISTEMA

Durante la fase de pruebas se pueden encontrar muchos errores que no fueron detectados en actividades previas a la verificación. Dos aspectos de la V&V se refieren a las pruebas del sistema. El primero de ellos, que es la función tradicional de la V&V, es la planeación independiente; conducir, analizar y reportar las pruebas por si mismas. Su objetivo permite practicar las capacidades que involucran el software del sistema.

El segundo aspecto es monitorear y revisar el avance de las pruebas del sistema. El staff de V&V tiene la capacidad de determinar el avance de las pruebas, proporcionando un informe completo para dar a conocer el grado en el que se ha desarrollado el software, mostrando una interpretación adecuada de los resultados de las pruebas, así como aquellas que han sido concluidas y cuáles de los reportes de las pruebas reflejan detalladamente y con exactitud los resultados de las pruebas. Las funciones de V&V asociadas con las pruebas incluyen:

- Verificación de que el sistema incluye los requerimientos y especificaciones establecidas para su desarrollo.
- Demostración de que la interfase entre el usuario y el sistema reúne expectativas (validar que los datos de entrada pueden producir los resultados requeridos en la salida, una entrada no válida puede ser manejada de manera prescrita sin causar defectos en el sistema).
- Generar la suficiente retroalimentación en cada etapa de desarrollo para prevenir grandes cambios en las actividades finales de implementación.
- Verificación de los módulos, bases de datos e interfases durante todas las actividades de integración del sistema.
- Preparación de todos los aspectos de pruebas de aceptación, incluyendo manuscritos o escenarios, documentación de los reportes y selección y preparación del equipo para soportar las pruebas.

Un grupo independiente de V&V es particularmente importante para las pruebas y evaluación del sistema.

No obstante, los programadores sondean sus diseños y codificaciones de programas para descubrir errores o deficiencias, pero estas pruebas son un tanto superficiales, ya que lo que les interesa es continuar con el siguiente programa para avanzar en su desarrollo lo más pronto posible.

5.12 NIVELES Y TIPOS DE PRUEBAS

Las pruebas deben comprender varias fases del ciclo de vida del software, por lo que estas pruebas se clasifican en varios niveles.

En las pruebas de unidad/módulo, las unidades individuales o módulos son depurados de manera aislada. Esta situación, generalmente es ejecutada por el

personal desarrollador del módulo, en donde la participación del staff de V&V es mínima.

Las pruebas de los subprogramas involucran: la integración de los módulos individuales en los subprogramas, la configuración de los programas de computadora, así como la demostración de las funciones que tienen los elementos de la configuración en los programas de computadora.

La integración y las pruebas del sistema, pueden ser ejecutadas por el staff desarrollador del mismo en coordinación con los grupos de prueba y de V&V.

Durante las pruebas de las funciones de los elementos de la configuración en un programa de computadora, se valida el apego de ésta con los requerimientos de la ejecución de un programa.

Las pruebas deben ser conducidas por un equipo independiente de pruebas y monitoreada por el staff de V&V. La terminación exitosa en las pruebas de las funciones de la configuración de un programa de computadora, puede garantizarse eliminando la mayor parte de los errores de programación y teniendo listo el software necesario para la integración del sistema.

Las pruebas de integración del sistema comprenden: la verificación de la integración de los programas con el equipo, otros sistemas y las interfases del usuario-sistema, así como también la funcionalidad y satisfacción de la ejecución de los requerimientos. El staff de V&V es responsable de la planeación, ejecución, análisis, generación de reportes y monitoreo de las pruebas del sistema.

Las pruebas de comprensión del sistema en su ambiente operacional en contraste con las especificaciones del mismo, establecen la aceptación de la aplicación de la calidad en la ejecución del sistema. Estas pruebas demuestran la disposición para la operación. Un grupo independiente, puede ejecutar estas pruebas en cualquier momento y con el presupuesto autorizado para ello.

A través de la aceptación de las pruebas podremos llegar al fin de ciclo de desarrollo del software, el administrador del sistema deberá iniciar la planeación

del mismo, durante el análisis de los requerimientos. El staff de V&V puede determinar cual de los requerimientos es susceptible de ser probado.

El tipo de pruebas usado para la aceptación del sistema puede variar. Los tipos más básicos son:
A) Los pruebas de sección, en los cuáles la estructura del programa es cuidadosamente examinada y cada instrucción o sección es probada por lo menos una vez.

B) Los pruebas funcionales: en las que el programa es tratado como una caja negra y probado directamente con los requerimientos del sistema.

La dependencia y las pruebas funcionales se complementan entre si. Las pruebas de dependencia obligan al programador a examinar cuidadosamente la estructura del código y de este modo a usarlo para llevar a cabo las pruebas de integración.

5.13 REVISION DE LAS PRUEBAS

El staff de V&V puede revisar el plan completo para la integración, incluyendo el plan de pruebas, procedimientos y reportes para verificar que éstos se apegan a los estándares establecidos para las pruebas. La documentación de las pruebas debe ser revisada manualmente, debido a que las herramientas no automatizadas para el análisis de los planes de prueba y procedimientos, generalmente no están disponibles.

La revisión puede ser ejecutada por un miembro del staff de V&V, el cual no se ha involucrado en el desarrollo de las pruebas de programación. El staff de V&V debe rastrear las violaciones a los estándares aprobados para las pruebas y la integración de los programas. Ejemplos de posibles violaciones son: el uso de métodos de integración abajo-arriba, supresión de las pruebas de regresión cuando el módulo mayor es añadido al sistema, integración del avance de las pruebas de unidad. El staff de V&V tiene la capacidad de identificar todas las ocurrencias de no conformacia de los procedimientos durante la ejecución y puede examinar todos los reportes para garantizar que los resultados son correctos.

5.14 PRUEBAS DE INTEGRACION

Las pruebas del sistema implican varias clases de actividades como son las pruebas de integración y las de aceptación entre otras.

Las estrategias para integrar los componentes del software en un producto que funcione incluyen las estratégica ascendentes y descendentes, así como las de emparedado.

Se requiere de una cuidadosa planeación y programación a tiempo para asegurar que los módulos estarán disponibles para su integración dentro del producto de software de desarrollo cuando se necesiten.

La estrategia de integración dicta el orden en que los módulos deben estar disponibles, por lo que ejerce gran influencia en el orden en que se escriben, depuran y hacen las pruebas de unidad de los módulos.

Las pruebas de aceptación implican la planeación y ejecución de pruebas funcionales, de desempeño y de tensión para verificar que el sistema realizados satisfaga sus requisitos.

Las pruebas de aceptación suelen realizarlas las organizaciones de control de calidad, los usuarios o ambos.

Una persona sin experiencia dentro del ambiente de la informática después de haber probado todos los módulos de un sistema por separado pensaría que ya probados estos no tendrán por que fallar en forma integral. En realidad este cuestionamiento es francamente lógico, pero la realidad es que la integración de todos los módulos no es una tarea fácil.

Una interfase puede estar diseñada en forma errónea, un módulo específico puede tener un efecto negativo sobre otro, un procedimiento puede comportarse

inadecuadamente por la forma en la que se le dan los datos de otro módulo, la impresión puede se enorme en conjunto, etc.

Para esto recurrimos a la prueba de integración que nos es mas que una técnica sistemática para construir la estructura

del programa mientras por otro lado se efectúan pruebas para tener detectadas las fallas que pueden influir en la integración. El objetivo es que podamos tomar los módulos que fueron probados en unidad y con ellos poder establecer una estructura acorde a el diseño original.

Un error que se comete a menudo es el querer integrar los módulos de forma anticipada, esto es que se prueba el programa en forma total.

Al realizar la pruebas en esta forma lo que se logra es que se tenga un caos ya que saldrán un sinfin de errores causando una difícil tarea de investigación para una detección adecuada de cada uno, y si lograrán detectar estos, lo mas probable es que saldrían unos nuevos.

Es necesario que se utilicen pruebas de integración incremental, con está se logra una integración real ya que se basa en la construcción y prueba de pequeños segmentos en donde se pueden detectar los errores fácilmente, así se logra que las interfases se integren con mayor fiabilidad.

5.14.1 INTEGRACION DESCENDENTE

La forma en la cual se maneja es mediante moverse hacia abajo dentro de la estructura jerárquica de control del programa. El primer modulo a ser analizado es el programa principal, los siguientes módulos a integra son aquellos que tienen una relación directa con este, así primero se integra el modulo principal en profundidad y luego en anchura.

En la figura 5.2 podemos observar que primero se integra en forma de profundidad dentro del camino de control del programa principal, el camino seleccionado esta dado en forma un tanto arbitraria ya que está depende mucho de la estructura de nuestro programa.

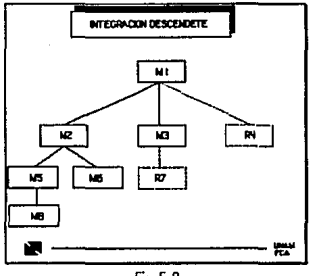


Fig 5.2

En el ejemplo si decidimos tomar el camino de la izquierda, los primeros módulos a integrar serán M1, M2 y M5, a continuación seguirá el M8, o (de ser requerido para que funcione adecuadamente M2) M6. Después se integran en forma de anchura, los primeros módulos integrados son el M2, M3 y M4 (remplazando el R4), y así seguirán los módulos M5 y M6.

Para efectuar esta integración se cuentan con una serie de pasos que se describe a continuación:

1. Se usa el módulo de control principal para que guía la prueba manteniendo siempre resguardos para todos los módulos que interactuen en forma directa del principal.
2. Dependiendo por donde empezamos la integración (ya sea en profundidad o en anchura) se sustituyen los resguardos subordinados uno a uno por los reales.
3. Cada vez que se integre un módulo se realizan pruebas.

4. Al terminar cada prueba se remplaza un resguardo mas con el módulo real.

5. Se efectúa una prueba de regresión ósea que se realizan las pruebas anteriores con el afán de verificar que no existan errores nuevos.

Esta estrategia de integración presenta las siguientes ventajas:

* La integración del sistema se distribuye en toda la fase de implantación. Los módulos se integran a medida que se desarrollan.

* Las interfases con el nivel mas alto se prueban primero y varias veces.

* Las rutina del nivel mas alto proporcionan datos de prueba naturales para las rutinas de niveles inferiores.

* Los errores se localizan en los nuevos módulos e interfases que se estén añadiendo.

Asi pues pareceria siempre deseable tener una integración descendente, pero en realidad existen casos en los que puede ser dificil encontrar datos de entrada del nivel principal o mas alto que ejerciten a algún modulo bajo en la forma en la que lo necesitamos.

La estrategia de integración descendente efectúa verificaciones a los puntos de control mas rápido en las fases de prueba. Asi, si contamos cómo una estructura jerárquica de proceso adecuada podremos contar que la toma de decisiones mas importantes deber estar en niveles altos y por lo tanto podremos verificar estos antes.

5.14.2 INTEGRACION ASCENDENTE

Esta prueba como su nombre lo dice es una prueba que comienza con los módulos atómicos, ósea los módulos mas bajos. Dado que los módulos se integra de abajo hacia arriba, el proceso de los módulos subordinados siempre esta presente y se elimina la necesidad de tener resguardos.

Para efectuar esta integración podemos seguir los siguientes pasos:

1. Combinar los módulos de bajo nivel en varios grupos que ya agrupados realicen una función determinada del software.
2. Se requiere construir un programa que controle las pruebas para que este coordine las entradas y salidas de los casos en prueba.
3. Se efectúa una prueba con el grupo.
4. Se eliminan los programas de control de pruebas y se efectúa una combinación de los grupos trasladandose hacia arriba por la estructura del programa.

En la figura 5.3 se hace una combinación de los módulos para formar 3 grupos el 1, 2 y 3 todos ellos se prueban bajo un programa de control que esta ilustrado en forma punteada. Los grupos 1 y 2 están subordinados al modulo MA. Es necesario decir que mientras mas arriba se corre en la estructura, es menor la necesidad de tener programas de control de pruebas.

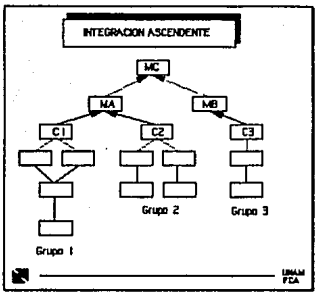


Fig. 5.3

5.15 PRUEBAS DE INTEGRACION ASCENDENTE CONTRA INTEGRACION DESCENDENTE

Se ha estado tratando a estas técnicas como técnicas separadas, aunque la mezcla de estas dos es posible, aun mas, es deseable. Tenemos varias secuencias de desarrollo puro de ambas estrategias en las fases de diseño, codificación y pruebas, en esta gráfica se asume que en cada una de estas fases es factible que se efectúen tanto en forma ascendente como en forma descendente, y su viabilidad.

Durante las fases de prueba una de las mayores diferencias entre la integración descendente y la integración ascendente es la necesidad de contar con resguardos como se vio en los puntos anteriores.

Como una situación practica podemos decir que no existe ninguna ventaja sin su correspondiente desventaja. Si nos decidimos por una codificación ascendente aplicandole una prueba de integración ascendente con el afán de eliminar los resguardos, tendremos que crear programas de control de pruebas, así estas dos técnicas son buenas pero es necesario efectuar un análisis profundo para determinar el mas adecuado a la estructura de nuestro desarrollo.

Como es de esperarse en el mundo real casi no existen estructuras puras ascendentes o descendentes, en realidad existen versiones modificadas o combinadas de ambas.

La primera aproximación seria la de generar una combinación de ambas en la que podamos contar con las partes buenas de una y otra técnica, así como eliminar sus problemas mas notables aunque sin perder de vista que esto también escapa a la realidad el administrador del sistema debe de encontrar las bondades de las dos sin crear mayores problemas y crear un caos en la mezcla.

Algunas de las aproximaciones más validas para esto las dió Myers y a continuación las exponemos:

Prueba modificada descendente. Una de las principales limitaciones de la técnica descendente es que no todos los módulos criticos pueden ser probados de inmediato, sino hasta que se termia de verificar toda la estructura, es por esto que se pensó en la necesidad de crear una modificación para que pueda funcionar de manera mas adecuada. Esta modificación consta en la misma técnica mas un incremento muy fuerte en las pruebas de unidad en los módulos criticos.

Pruebas Ascendentes-Descendentes: Si existe más de un módulo critico, nos veremos forzados a considerar el trabajar desde las los puntas (arriba y abajo) de la estructura, recordando que lo importante es que se encuentren a la mitad de la estructura. También de gran importancia es escoger a que módulos conviene trabajarlos en forma descendente y cuales en forma ascendente. Recordemos que los módulos criticos se deberán manejar en forma ascendente. Tomemos en cuenta que esta forma de trabajar es delicada, ya que si estamos deseando tener las bondades de las dos técnicas podriamos tener También todos los problemas de ellas.

Pruebas de regresión: En las técnicas de integración cuando encontramos un error lo corregimos y prosedemos a regresar y a repetir la prueba. Esta repetición ya se en forma completa o parcial es conocida como Regresión, es el caso de muchos autores que dan demasiada importancia a los costos y que advierten de el tratar de evitar estas regresiones, independientemente de esto, es prudente el que realicen estas pruebas que aunque repetitivas nos dan la ventaja de garantizar nuestro desarrollo informático.

5.16 PRUEBA DE VALIDACION

Ya culminada la fase de las pruebas de integración podemos decir que el software desarrollado se encuentra unido en un paquete, en el cual ya se detectaron los errores y fueron corregidos inclusive en sus interfases.

Para entender que significan estas pruebas podemos decir que estas se refieren a que el desarrollo realizado efectué lo que el usuario o cliente espera. Claro que estas espectativas del usuario debieron quedar antes bien detalladas en un documento conocido como la: Especificación de Requerimientos de Software.

Para que se de la conformidad del usuario se hacen una serie de pruebas de tipo caja negra en la cual se le entregan resultados tangibles y entendibles acordes a sus espectativas. Se hace un plan de pruebas y un procedimiento de pruebas mismos que estarán diseñados para asegurar que satisfacen todos los requerimientos funcionales para la correcta operación del sistema.

Como consecuencia de este paso se pueden despegar solo dos posibles situaciones. 1) que se confirme que el sistema esta funcionando como debería de funcionar, o 2) que en las pruebas finales se detectaron fallas minimas, mismas que son muy dificiles de corregir a esta altura del proyecto, es entonces cuando el o los administradores del sistema deberán de negociar estas fallas con el usuario para que se pueda llegar a un acuerdo satisfactorio y realizar una planeación para optimizaciones posteriores.

Es conveniente el que demos una repasada en este punto sobre la estructura que seguimos durante el desarrollo del sistema, ya que con esto sabremos que en realidad todos los elementos del sistema fueron desarrollados de forma adecuada. fig. 5.4.

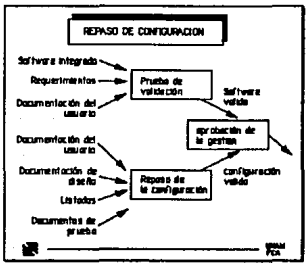


Fig. 5.4

Para el equipo de desarrollo es muy difícil el preever que el usuario va a dar una utilidad adecuada del software entregado, esto se debe a que el usuario puede dar interpretaciones incorrectas a las instrucciones que se le entregan en los manuales de operación.
Es por esto que cuando el desarrollo fue terminado enfocado a las necesidades especificas de nuestro usuario es de vital importancia que este realice las pruebas de aceptación del sistema y así validar en realidad que están cubiertas sus necesidades.
Esta prueba puede ser tan sencilla como una simple presentación en vivo, o tan complejo como son una serie de pruebas que pueden inclusive durar algunas semanas, esto acarreara la detección de muchas pequeñas imperfecciones del sistema mismas que de alguna forma son normales ya que ningún sistema es del todo perfecto.

Si el software se desarrollo para varias áreas dentro de la empresa, o inclusive esta enfocada a diferentes empresas no es conveniente efectuar pruebas de aceptación a todos ellos, es cuando aplicamos una prueba conocida como Prueba alfa beta,
esta prueba consta en que en el lugar de desarrollo se da cita a una de las áreas usuarias para que estén viendo como el desarrollador se encuentra efectuando pruebas del sistema, a esto se le conoce como prueba alfa, por otro lado en el resto de las áreas usuarias se les da una copia del mismo sistema y efectúan pruebas en vivo, a esto se le conoce como pruebas beta.

5.17 PRUEBA DEL SISTEMA

El hecho de haber efectuado todas las pruebas anteriores no es una garantia definitiva para que el desarrollo final funcione, pero si es de alguna forma una buena aproximación de lo que se necesita.

Es común que se encuentre que cuando aparece un problema en las pruebas no se pueda determinar con veracidad las pruebas del sistema, ya que cada uno de los integrantes del desarrollo se echaran la culpa dificultando la detección real del problema.

Por esta razón el encargado del desarrollo del sistema deberá de seguir los siguientes pasos para preever que no se presentes estos problemas:

- Diseño de caminos para manejo de errores.
- Realizar pruebas con datos defectuosos.
- Realizar una bitácora de estas pruebas.
- Participar directamente en el diseño de las pruebas integrales del sistema.

La prueba del sistema esta integrada por una serie de pruebas diferentes que ejercitan finalmente al sistema en forma tal que podemos observar si se ha echo una fusión adecuada de las partes del sistema, las pruebas que se manejan dentro de esta sección son:

Pruebas de recuperación: Es conveniente que los sistemas estén diseñados de tal forma que sean capaces de manejar las fallas que puedan presentarse, esto no implica que se tuvieran dadas las soluciones a todas las posibles fallas, sino que el sistema deberá de ser capaz de informar al usuario de que existe alguna falla en la operación del sistema y no que de forma total se detenga el sistema ya que esto causa serios daños en el plano económico para la empresa.

Esta prueba esta diseñada para enfrentar al sistema a una serie de fallas que se pretenden sean la mayoria posible, para estos efectos debemos de considerar que las fallas se podran resolver de dos formas:

a) Automáticamente
b) Manualmente

En ambos casos es conveniente que se evalué los posibles efectos en tiempo y en operación para obtener mejores resultados.

Prueba de seguridad: En la actualidad es de todos conocido que existe un gran problema en cuanto a la violación de la seguridad de los sistemas ya que se han desarrollado infinidad de virus que dañan la integridad de los programas, aunque también es real que este problema se limita en la mayoria de los casos a las microcomputadoras. El problema que existe con los equipos mini y macro se refiere en mayor proporción a los accesos ilegales en que incurre personal propio o ajeno a la empresa con el fin de alterar la información en las aplicaciones instaladas, estas incursiones ilegales están causadas por venganzas de empleados, por gente que lo hace solo por diversión, etc.

El fin de implantar una prueba de seguridad a nuestro desarrollo es justamente para verificar que tenga las suficientes protecciones para no permitir el acceso al sistema. Se le asigna a una persona el papel de un individuo que pretenda accesar el sistema y no se le limita de ninguna forma para que por cualquier medio intente entrar al software diseñado. Al final de las pruebas el desarrollo estará lo suficiente protegido en contra de la mayoria de accesos ilegales. Por tanto para desanimar a cualquier persona de entrar sin autorización se pretende que sea mayor el costo del acceso que el mismo valor de la información.

Prueba de resistencia: Esta prueba esta diseñada para que se vea cuanto puede soportar de carga un sistema, ya que en muchos casos el sistema ya fue probado y en la realidad cuando se carga el sistema con datos voluminosos no le es posible operar adecuadamente.

La prueba no es mas que intentar sobrecargar el sistema de interrupciones, datos, frecuencias de estos, búsquedas excesivas, etc. o sea que el encargado de la prueba tratara de tirar el sistema, en el caso de desarrollos que estén enfocados a dar servicio a usuarios esta prueba es de mucha ayuda ya que se podra medir con mas veracidad la estabilidad del sistema en casos extremosos.

5.18 REVISION DE LA CONFIGURACION

El staff de V&V ejecuta algunas funciones importantes durante la revisión de la configuración. Estas funciones son soportadas por el staff administrador de la configuración, el cual tiene como responsabilidad primaria el revisar y ejecutar sus funciones de verificación para determinar si existe adherencia con los estándares y convenciones establecidos, usando formatos especiales para expresar sus comentarios y dar a conocer las discrepancias encontradas. El staff de V&V puede investigar los comentarios expresados, pero la acción correctiva no necesariamente se tiene que llevar a cabo para completar satisfactoriamente la revisión. Una discrepancia, muestra que se ha identificado inadecuadamente o que no se ha cumplido con las convenciones y estándares establecidos. Esta situación puede mostrar claramente la diferencia entre las condiciones actuales y las esperadas. En la medida en que las discrepancias encontradas sean corregidas, se completará la revisión.

Dos revisiones formales de la configuración ocurren muy al final de la fase de desarrollo del software, antes de la fase de operación y mantenimiento para verificar entre otras cosas la revisión de la configuración física y la examinación formal del código contra la documentación técnica. Las actividades de V&V para cada una de esas revisiones son similares, aunque son revisados de manera diferente.

5.19 REVISION FUNCIONAL DE LA CONFIGURACION

Antes de realizar esta revisión, el staff de desarrollo debe proporcionar una lista de los elementos a ser auditados. Paralelamente a la revisión, se verifica esta lista con el fin de ver si todos los elementos a revisar se encuentran disponibles, completos y actualizados. El staff de V&V verifica la documentación actual (incluyendo manuales de usuario) para asegurar que se ha completado y es entonces, cuando se prepara la lista de documentación y programas que deben estar disponibles para la revisión de la configuración física.

5.20 REVISION DE LA CONFIGURACION FISICA

El staff de V&V debe confirmar que los listados actuales son correctos y estan disponibles para cada componente de programa de computadora. El mantenimiento de los datos es revisado para garantizar que son correctos en tipo y cantidad. Los siguientes elementos deben ser verificados para que las entradas y los formatos usados sean apropiados:

- Especificaciones de diseño de programas
- Descripción de los componentes de los programas de computadora y de los requerimientos para las interfases
- Diagramas del flujo funcional
- Descripciones de las bases de datos
- Localización del almacenamiento de caracteres
- Guias y manuales
- Documentación de la descripción de programas

5.21 OPERACION Y MANTENIMIENTO

La necesidad de la V&V continúa en las fases de operación y mantenimiento del desarrollo del sistema. La V&V tiene el propósito de confirmar la liberación y aceptación del sistema (software desarrollado), para conformar las especificaciones establecidas y ejecutar satisfactoriamente las funciones para las cuales fuè diseñado. Sin embargo, el nivel de las actividades de V&V cambian durante la operación y mantenimiento. Antes que el direccionamiento de la actividad continua, la V&V comprende dos actividades que ocurren en diferentes puntos. La primera involucra la identificación de los errores y la verificación de que éstos son corregidos. La segunda incluye la verificación y validación de los

cambios hechos al software. Los requerimientos del staff durante esta fase reflejan la naturaleza de las actividades de V&V. Sin embargo, la necesidad de la V&V durante la operación y mantenimiento del sistema es indispensable.

El uso operacional del software, continuamente resalta los errores que no fueron descubiertos antes de la aceptación; probar el sistema completamente, es una tarea casi imposible. La documentación errónea o inadecuada es una fuente de errores, debido a que el software no es lo que se esperaba, como lo expresaba la documentación. Cuando un error es detectado, el reporte de problemas de software es archivado para llevar a cabo una acción. El grupo de V&V es responsable de identificar las causas de los errores, pero el staff de operación y mantenimiento es, generalmente el responsable de corregir esos errores. Una vez que el error ha sido corregido, el grupo de V&V debe probar la parte del software corregido y conducir a las pruebas de regresión, para garantizar que el sistema ha sido ejecutado completamente como se esperaba en su descripción. Algunas veces, el descubrimiento de errores nos conduce a mayores cambios en el sistema, debido a que el error no puede ser corregido sin modificar el sistema. Los cambios pueden mejorar la ejecución o eliminar una fuente de errores. Los cambios tanto en hardware como en software, incluyen: software mejorado, .modificaciones al software actual y propuestas de cambio en la ingenieria de software. El grupo de V&V revisa el impacto del análisis de los cambios propuestos antes de aprobarlos para su realización. Si el cambio debe ser implementado, la V&V se involucra para verificar que el diseño esta de acuerdo con los cambios especificados y que la implementación refleja correctamente el diseño. La V&V es usada para la planeación de las pruebas, ejecución de las pruebas para los cambios de las partes del software que será modificado y la ejecución de las pruebas de regresión del sistema. Esto es una gran responsabilidad, que debe dar a conocer la certeza de los cambios realizados al software y garantizar que la documentación refleja todos los cambios.

5.22 CURSO DE ACCION RECOMENDADO

No obstante que las actividades de V&V sobreponen la función de aseguramiento de la calidad, se debe dar un gran énfasis a la V&V en cuanto a su precisión técnica de acuerdo a los estándares y convenciones establecidos. Cuando el software es correctamente aplicado, la V&V puede determinar que los requerimientos que se dan al sistema se encuentran en constante actualización, del mismo modo permite saber si el software se ha desarrollado apegándose a las especificaciones establecidas y que la ejecución de las funciones para lo cual fueron diseñadas, es satisfactoria.

5.23 AUDITORIA EN SISTEMAS

Hoy en dia la mayoria de las organizaciones de medianas a grandes invierten una fuerte suma de recursos en establecer controles adecuados para la organización y para sus sistemas de computo, para que esto sea efectivo un primer paso es el de efectuar auditorias en sistemas.

Dentro de la auditoria en sistemas tomaremos en cuenta los puntos que nos indica Richard Fairley:

1. Auditoria de los procedimientos dentro de los centros de computo:
+ Organigrama de los miembros del centro.
+ Separación de funciones entre los miembros; verificación contra el control.
+ Separación de funciones entre la programación y la operación.
+ Mantenimiento de bitácoras de control para la entrada y la salida.
+ Calendario de trabajo regulares.
+ Procedimientos de respaldo de archivos; almacenamiento

de copias de seguridad en otro loca.
+ Procedimientos de respaldo de programas; almacenamiento de copias (incluso documentación) de seguridad en lugares fuera del mismo centro.
+ Procedimientos para revisar los cambios de programas.
+ Biblioteca y comprobación de programas así como bitacora de cambios de programas.
+ Controles de la biblioteca de discos y cintas.
+ Etiquetas externas de archivos.

2. Guías para la auditoria de telecomunicaciones.
+ Codificación:
    - Necesidad de codificación.
    - Acceso a las claves.
    - Algoritmo utilizado.
+ Detección de errores.
+ Corrección de errores.
+ Retransmisión.
+ Confiabilidad de la red.
+ Respaldo de la red.
+ Programas de aplicación:
    - Revisión de errores.
    - Revisión de datos completos.
    - Respaldo.
    - Bitácora de mensajes.

3. Guías para la auditoria de procesos:

+ Archivos
    - Revisión de secuencia
    - Revisión de congruencia de los campos de entrada
    - Revisión de congruencia en la actualización.

- Conteo de registros totales parciales, etc.
- Relación de los registros secundarios con los totales obtenidos para revisar la integridad.
- Mensajes de error apropiados para transacciones no encontradas.

+ Uso de etiquetas de archivos y revisión de errores en todas las etiquetas de las corridas.
+ Revisión de la secuencia en la ejecución de los programas.
+ Balanceos de ajuste y revisiones de las salidas.
+ Registros de auditoria, especialmente para procesos en linea.
+ Posibilidad para rastrear todas las transacciones del sistema.
+ Posibilidades de respaldo y de reinicio para procesos largos.
+ Registro completo de los cambios efectuados a los archivos.

4. Guia para la auditoria de entrada/salida.

+ Proceso manual:
  - Verificación que todas las transacciones se han recibido para su proceso..
  - Incentivos a las fuentes para que los datos sean correctos.
  - Autorización para actualizar la base de datos.
  - Revisión de la transmisión de datos.
  - Revisión de la conversión de datos.
  - Procedimientos para cuando no exista concordancia en los datos de entrada.
  - Funciones de prueba y control realizadas por otro que no sea el operador de la máquina.

- Seguimiento a la corrección de errores.
- Calendario de las salidas para los usuarios
- Revisión de los informes de salida.

+ Proceso de máquina.
  - Control de entrada:
    x Registro de control en lotes, cálculos duplicados por el computador.
    x Edición de campos.
    x Digitos de paridad.
    x Revisión de datos faltantes.
    x Prevenir el proceso duplicado de entradas.
  - Control de salida:
    x Conteo de registros sobre los reportes de salida.
    x Totales de control.

5. Guías para la auditoría de documentación.

+ Lista de aplicaciones.
+ Diseño y mantenimiento de los sistemas.
  - Tabla de contenido
  - Estudio de factibilidad.
+ Sistema actual.
+ Especificación del nuevo sistema.
  - Salida.
  - Entrada (formas y códigos actuales).
  - Descripción y organización del archivo.
+ Módulos de los programas.
+ Diagramas de flujo de sistemas.
+ Listado de cada versión.
+ Diagrama de flujo/tablas de decisión.
  - Lista de referencia cruzada de las variables
  - Referencia cruzada de la interrelación de

Módulos.
- Variables e identificadores.

+ Pruebas
  - Diseño / Datos / Resultados

+ Procedimientos manuales.
  - Diagramas de flujo.
  - Descripción.
  - Control de errores.
+ Planes de trabajo.
  - Informes de avance.
+ Documentación del usuario.
  - Salida (cómo interpretarla)
  - Entrada (cómo complementarla y suministrarla).
  - Archivo.
  - Procedimientos de proceso.
  - Errores.
  - Consideraciones de transacción.
  - Conversión.
+ Documentación del operador.
  - Diagramas de flujo del sistema.
  - Lista de programas.
  - Para cada programa.
    x Entrada y formato.
    x Archivos.
    x Descripción del proceso
    x Salida producida.
    x Condiciones de errores y acciones a tomar.
    x Reinicio.
+ Distribución y proceso de los informes.
+ Programador responsable del sistema.
+ Usuario responsable del sistema.
+ Corte en el suministro de entrada.

+ Calendario de los tiempos de ejecución así como la disponibilidad.
+ Requerimientos de preparación.
+ Lista de los componentes de máquina que se usan.

CONCLUSIONES

El control de calidad en el proceso de desarrollo de software es una de las principales caracteristicas que debemos tomar en cuenta para cumplir las necesidades del usuario. este concepto debe manejarse a todos los niveles de la organización.

El desarrollo en el personal, se basa en una constante educación mediante la planeación de capacitaciones que permiten la eficiencia y motivación en el desempeño de sus funciones que redunda en la obtención de mejores resultados para la empresa y el cliente.

El establecimiento de una estructura organizacional funcional, permite mejorar el flujo de la información, ya que a medida que esta comunicación es eficiente, permitirá inducir el concepto de calidad a todos los niveles, generando en consecuencia la obtención de los resultados esperados.

La búsqueda continua del control y mejoramiento de la calidad en todas las etapas del desarrollo de un sistema, son la base para la productividad y competitividad de una empresa.

La esencia de este enfoque radica en el manejo y aplicación adecuados de la herramientas técnicas y estándares, asi como un buen control de proyectos que incluye: estimación de los niveles de contratación para los proyectos, control presupuestal, estimación de costos e involucramiento de los recursos humanos incluyendo a los usuarios y no como tradicionalmente sucede por efecto de nuestra cultura, depender exclusivamente de la experiencia e intuición.

No debemos olvidar el establecimiento de mecanismos de control tanto en la etapas de desarrollo (pruebas modulares) como al final del mismo (pruebas integrales) que nos permitan verificar y validar la calidad y funcionalidad del sistema.

Los principales objetivos de la verificación y la validación son evaluar y mejorar la calidad de los distintos productos de trabajo, generados durante el desarrollo y/o la modificación del software.

Mejorar continuamente la calidad, aumentar y fomentar la productividad, así como fortalecer la posición competitiva son prpositos de cualquier tipo de empresea actual, ya que del cumplimiento de esos propósitos puede depender la supervivencia de la organización en el ámbito empresarial de nuestros días.
La alta calidad no se puede lograr dentro de un sistema que ha sido creado con equivocaciones y mal implantado.

De lo anteriormente escrito se concluye:
Aplicar el concepto de calidad así como todo lo que este implica es una difícil labor que demanda un gran esfuerzo, dedicación y un compromiso del personal con su empresa, con su cliente, con su producto y/o servicio e inclusive con su país.

Los dueños o directivos de las empresas ya no solo deben pensar en sus productos, sino que también deben poner mucha atención a su personal para así motivarlos, encaminando todos sus esfuerzos a la excelencia.
Analizar, definir y entender correctamente un proyecto nos brinda la posibilidad de poder automatizarlo. De lo contrario, aunque se utilicen técnicas o métodos muy sofisticados y avanzados nuestros esfuerzos no cumplirán con las necesidades del usuario.

Dentro de las principales funciones del grupo responsable del desarrollo de sistemas esta el controlar con presición todas las variables que estan a su alcance para evitar que se afecte el buen funcionamiento de su proyecto.

No obstante la aplicación de esquemas de calidad en las distintas etapas de desarrollo, se requiere verificar y validar el producto para evitar el riesgo de incurrir en errores en el producto final contribuyendo en el desacreditamiento de la empresa.

En México, las empresas requieren incrementar el nivel de capacitación integral a sus empleados ya que nos enfrentamos ante una apertura económica internacional que nos exige mejorar nuestros estándares de calidad.

Esta tesis es de carácter aplicativo y será funcional si consideramos la correcta aplicación de los conceptos utilizados en ella.

Bibliografía

LIONET, Patrik. Los Métodos de la calidad total. ESPAÑA, Ediciones Diaz de Santos, 1989, 213 pp.

McKEOWN, Davis. Modélos cuantitativos para Administración. MEXICO, Editorial Iberoamericána, 1986, 757 pp.

HAMPTON, David. Administración contemporánea. MEXICO, McGraww-Hill, 1981 ,579 pp.

SHOOMAN, Martin. Software Engineering, USA, International Edition Student, 1983, 683 pp.

FAIRLEY, Richard. Ingenieria de Software. MEXICO, McGraw-Hill, 1988, 390 pp.

PRESSMAN, Roger. Ingenieria de Software, un enfoque práctico. ESPAÑA, McGraw-Hill, 1989, 628 pp.

SENN, James, Análisis y diseño de sistemas de información. MEXICO, McGraw-Hill,1984, 641 pp.

GITLOW, Howard. Como mejorar la calidad y la productividad con el método Deming. COLOMBIA, 1987.

ISHIKAWA, Kaoru. Total Quality Control. USA. Prentice Hall, 1989.

BIGGS, Charles. The system development process. USA. Prentice Hall, 1987.

TERRY, Edward. Software Quality Assurance. USA. Software Engineerin Institute. 1990.

MEREDITH, Denis. Software Testing-Management Methods and Tools. USA. Software Engineering Institute, 1990.

PERRY, William. The role of verification and Validation in Software Quality Assurance. USA. Quality Assurance Institute Orlando, FL. 1989.