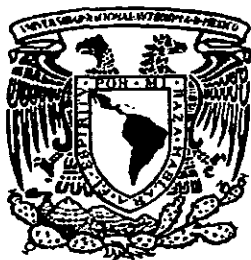


201

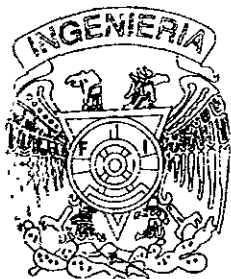


# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

## "METODOLOGÍA PARA EL DISEÑO DE TÉCNICAS DE PRUEBA DURANTE EL CICLO DE VIDA DE LOS SISTEMAS DE INFORMACIÓN"

T E S I S  
Que para obtener el título de  
INGENIERO EN COMPUTACIÓN  
p r e s e n t a  
Francisco Germán Rivera Téllez



Director de Tesis: Ing. Salvador Pérez Viramontes

México, D. F. Enero 1998

TESIS CON  
FALLA DE ORIGEN

262779



Universidad Nacional  
Autónoma de México



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

**A mi director de tesis, Ing. Salvador Pérez Viramontes, quien siempre ha sido un gran ejemplo de profesionalismo, constancia y tenacidad y quien además ha depositado una enorme confianza en este esfuerzo y en mi persona.**

**A mi madre, quien con su labor constante de apoyo, siempre me exhortó a concluir este proyecto de mi vida profesional. ¡ Mil Gracias !**

**METODOLOGÍA PARA EL DISEÑO DE TÉCNICAS DE PRUEBA DURANTE  
EL CICLO DE VIDA DE SISTEMAS DE INFORMACIÓN**

**ÍNDICE**

**DEFINICIÓN**

	<i><u>Página</u></i>
<b>CAPÍTULO I.....</b>	<b>1.</b>
<b>Importancia de las pruebas en los sistemas de Información.....</b>	<b>1.</b>
• <b>Introducción.....</b>	<b>1.</b>
• <b>El mercado demanda altos estándares de calidad.....</b>	<b>3.</b>
• <b>La economía se rige por la calidad.....</b>	<b>3.</b>

**MÉTODO**

<b>CAPÍTULO II.....</b>	<b>6.</b>
<b>Actividades mínimas de verificación y validación.....</b>	<b>6.</b>
• <b>Introducción.....</b>	<b>6.</b>
• <b>Etapas de análisis y definición.....</b>	<b>7.</b>
• <b>Análisis de riesgo.....</b>	<b>9.</b>
• <b>Control de cambios.....</b>	<b>13.</b>
• <b>Desarrollo de planes de prueba.....</b>	<b>15.</b>
• <b>Fase de diseño de negocios y diseño técnico.....</b>	<b>20.</b>
• <b>Planeación de las pruebas de integración.....</b>	<b>20.</b>
• <b>Fase de construcción.....</b>	<b>27.</b>
• <b>Pruebas unitarias.....</b>	<b>28.</b>
• <b>Pruebas de componentes No-Programa.....</b>	<b>30.</b>

<b>CAPÍTULO III.....</b>	<b>32.</b>
<b>Actividades opcionales de verificación y validación.....</b>	<b>32.</b>
• <b>Registro , seguimiento y resolución de problemas.....</b>	<b>32.</b>
• <b>Lista de categoría de defectos.....</b>	<b>34.</b>
• <b>Análisis de defectos.....</b>	<b>35.</b>
• <b>Independencia de pruebas.....</b>	<b>36.</b>
• <b>Pruebas Dinámicas.....</b>	<b>38.</b>
• <b>Material de entrenamiento.....</b>	<b>38.</b>
• <b>Manuales de usuario y operación.....</b>	<b>38.</b>
• <b>Manual de referencia.....</b>	<b>39.</b>

## APLICACIÓN

<b>CAPÍTULO IV.....</b>	<b>40.</b>
<b>Criterios para establecer un conjunto de pruebas para un sistema.....</b>	<b>40.</b>
• <b>Introducción.....</b>	<b>40.</b>
• <b>Procedimientos de verificación.....</b>	<b>40.</b>
• <b>Autoevaluación.....</b>	<b>41.</b>
• <b>Pruebas estáticas.....</b>	<b>42.</b>
• <b>Pruebas dinámicas.....</b>	<b>44.</b>
• <b>Criterios para establecer un conjunto de pruebas.....</b>	<b>44.</b>
• <b>Pasos para determinar los conjuntos de pruebas.....</b>	<b>44.</b>
• <b>Proceso de desarrollo de la estrategia de pruebas.....</b>	<b>45.</b>
• <b>Puntos de control y atributos que son controlados.....</b>	<b>50.</b>
• <b>Pruebas en el ciclo de vida de los sistemas.....</b>	<b>51.</b>

**CAPÍTULO V..... 52.**

**Herramientas y técnicas de prueba..... 52.**

- Técnicas de tiroteo “Shotgun”..... 52.
- Técnicas de prueba basadas en los requerimientos..... 53.
- Técnica de Causa-Efecto..... 53.
- Técnica de partición de equivalencias y análisis de valores límite..... 57.
- Técnica de adivinar errores..... 59.
- Pruebas basadas en la lógica..... 60.
- Gráficas para el control del flujo..... 60.
- Análisis de cobertura..... 65.
- Cobertura de una ruta o camino..... 71.
- Complejidad..... 71.
- Métodos para el cálculo de la complejidad..... 72.
- Determinación de caminos de cobertura..... 75.
- Elementos de la documentación de casos de prueba..... 77.
- Muestra de la forma para el examen de descripción del caso..... 78.

**CAPÍTULO VI..... 79.**

**Caso práctico de aplicación de técnicas de prueba..... 79.**

- Descripción del plan de proyecto para el sistema de compra de vehículos.. 79.
- Diagrama de contexto..... 80.
- Lista de eventos para el sistema de compra de vehículos..... 81.
- Diagrama de estructura..... 83.
- Definición de causa y efectos..... 84.
- Tablas de decisión..... 87.
- Casos de prueba ..... 88.
- Determinación de casos de prueba posibles..... 96

**RESULTADOS Y CONCLUSIONES**

**CAPÍTULO VII..... 97.**

- Resultados..... 97.
- Análisis de resultados..... 103.
- Conclusiones..... 108.
- Bibliografía..... 112.

## CAPÍTULO I

### IMPORTANCIA DE LAS PRUEBAS EN LOS SISTEMAS DE INFORMACIÓN

#### **Introducción.**

Esta tesis tiene el objetivo de crear una metodología que apoye el diseño e instrumentación de sistemas de información en una de las fases más importantes: la etapa de pruebas de funcionalidad y operación.

Esta tesis tiene el valor de crear estándares tanto en método como en forma para diseñar mejores planes de prueba y su aplicación al diseño de sistemas, encontrando el punto de equilibrio que maximice la rentabilidad y eficiencia del sistema, aunado a la satisfacción del usuario final.

Esta tesis propone una nueva forma de instrumentar sistemas desde la fase de diseño hasta la fase de implantación y soporte en la producción, considerando no solo al código fuente sino también todos aquellos elementos que apoyan y contribuyen al éxito de un nuevo sistema y su futuro mantenimiento y operación.

*Considerando que es muy duro y difícil cambiar la manera de hacer las cosas cuando estas se han venido haciendo de la misma forma durante toda la vida, pero en el mundo actual, los cambios en las demandas de los consumidores y la existente competencia obligan a los profesionales, las compañías y a países enteros a cambiar el hecho de solo estar presente, por una actitud de mejora continua.*

*Muchas compañías están encontrándose en que tienen que hacer las cosas de una manera diferente para poder sobrevivir en el mercado de hoy. La competencia mundial, reforzada por un nuevo y más inteligente estilo de administración, nunca ha sido más intensa. Las economías están transformándose ante la vista de cualquiera, y es posible percibir como grandes países y en consecuencia importantes empresas, están siendo reestructuradas y reformadas para mantenerse en competencia.*

*Los administradores están aprendiendo y experimentando nuevas maneras para dirigir a las compañías. Los empleados y trabajadores están aprendiendo cómo contribuir con su conocimiento y experiencia al mejoramiento de los procesos de producción, de servicio y en general, de todas las actividades implicadas en una organización. Se busca con gran entusiasmo el iniciar, mantener y acrecentar industrias y empresas más saludables en términos de largo plazo; y no solo obtener utilidades y beneficios en corto tiempo.*

Todos están escuchando con mayor atención a los clientes y usuarios finales, y en forma más efectiva, logrando que sus productos y servicios continúen siendo útiles y valorados. Si ellos no lo hacen, alguien más tomará la ventaja y captará a los usuarios, pues sin ellos no hay compañías.

Este nuevo enfoque de administración permite a las organizaciones mantenerse a la cabeza de estos rápidos cambios. Esta administración es practicada por la mayoría de las empresas japonesas que exitosamente sobrepasaron a las industrias clave de los Estados Unidos de Norteamérica en las últimas tres décadas. Estas compañías estadounidenses están comenzando a aprender y entender estos principios de administración, poniéndolos en práctica y retomando el reto.

Esta tesis considera a este nuevo estilo de administración como un LIDERAZGO DE CALIDAD.

Por el aprendizaje de cómo monitorear, controlar y constantemente mejorar la creación de sistemas de información, las organizaciones son más capaces y aptas para proporcionar a sus clientes lo que estos necesitan, cuando y como ellos lo quieren. Esta manera de hacer las cosas permite tomar mejores decisiones tanto a los clientes y usuarios, como a las organizaciones, trabajadores y administradores.

De esta manera, las decisiones están basadas en datos, y no en suposiciones o incluso adivinanzas.

El objetivo es el mejoramiento de los productos o servicios por la mejora en la manera en que se trabaja, es decir, los métodos con los que el trabajo es hecho, con la única idea de simplificar los resultados a través de los procesos.

Las relaciones entre los empleados y la administración deben ser reestructuradas; un gerente o administrador debe ayudar a la gente a hacer el trabajo lo mejor posible, eliminando las barreras que pudieran existir, para ayudar a producir en forma constante servicios de alta calidad.

***El mundo de la ingeniería de sistemas y computación no es la excepción, pues esta es una herramienta indispensable para la toma correcta y oportuna de decisiones. Este manual propone una serie de técnicas que apoyan y dirigen el mejoramiento del desarrollo de sistemas de información, a través de métodos y guías que aseguran el éxito de cualquier nuevo sistema de procesamiento de información.***

***El principal objetivo no solo es elaborar y diseñar sistemas más inteligentes, sino realmente cubrir las necesidades por las cuales fue creado y que además proporciona: un valor agregado que lo haga más útil y productivo, sin que por ello deba ser más caro.***



## **El mercado demanda altos estándares de calidad**

Actualmente se tiene significativamente más competencia hoy que en cualquier otro tiempo y esta irá en aumento a medida en que la gente tome conciencia de la importancia y los beneficios de ella. Los productos o servicios con calidad son esenciales para poder competir y asegurar una continua sobrevivencia de cualquier industria o actividad económica, y en ello juega crucial importancia la administración de la información, la cual estará soportada en sistemas computarizados que permitan un mejor uso de los recursos de una organización.

La retención de clientes actuales y la adquisición de nuevos dependerá únicamente de lo que ellos quieren y necesitan de aquellos que buscan soluciones para cubrir esas necesidades.

Los sistemas de información de hoy deben proporcionar datos estratégicos y tácticos, además de automatizar actividades; sin olvidar la importancia que tiene el compartir datos que sean comunes por diferentes grupos e individuos.

La probabilidad de crear un producto con cero defectos decrece proporcionalmente conforme a la complejidad y el número de procesos que participan crece. Consecuentemente aquellas compañías y países que no consideran a la calidad como una de las más altas prioridades a seguir, no podrán mantener una posición en el mercado ni sobrevivir. Por tanto deben producirle productos de excelente calidad; pues hoy por hoy "todos quieren calidad".

## **La economía actual se rige por la calidad**

La filosofía que debe ser seguida por aquellas empresas y organizaciones que provean o proporcionen servicios de información, deberá regirse bajo el siguiente principio:

"Proveer un producto o servicio de la más alta calidad, que pueda crear un gran valor para los clientes; de tal manera que se puedan mantener relaciones duraderas caracterizadas por el mutuo beneficio". Esto depende y está fundamentalmente basado en las actividades que día a día realizan los empleados, es por eso que la calidad debe ser integrada en nuestras actividades diarias y adoptarla como un estilo de vida, pues todos somos responsables de ella.

El compromiso hacia los clientes, la calidad, el crecimiento y el beneficio global, es el compromiso que conlleva a crear una metodología de sistemas, diseñada para facilitar la integración de los sistemas, sin olvidar el costo-beneficio que esto representa.

La prueba exhaustiva de los sistemas de información, garantizará los resultados esperados, lo que redundará en un producto o servicio final de alta calidad y libre de defectos.

---

Hasta hoy se consideraba que los sistemas de información solo debían ser probados en la fase de construcción, en la que el código es generado. Sin embargo, esto ha resultado en innumerables ocasiones en sistemas altamente costosos, ineficientes y que rápidamente se vuelvan obsoletos, además de que antepone una gran dificultad para integrarse a una solución corporativa.

Es por ello que debe tomarse una diferente actitud: ver a las pruebas no solo como una fase o paso importante en la construcción, sino como todo un proceso a través del ciclo de vida de cualquier sistema.

Esto viene a ser confirmado cada vez que se presentan los fracasos en el intento de querer concluir un sistema, probarlo o simplemente usarlo. En innumerables ocasiones se ha sido testigo de experiencias en las que un sistema solo fue probado en su codificación. Se tuvo como objetivo único el lograr que una serie de programas y rutinas funcionara con ningún código de error; pero los resultados son contundentes: el sistema no genera lo esperado, porque nunca se validó la idea original, el concepto fundamental y los obvios, y nunca se cuestionó si era factible o rentable.

El evitar caer en estas circunstancias es prácticamente imposible si no se cuenta con una técnica y una disciplina que permita, que al generarse la más mínima unidad de información, esta pueda ser probada, de tal forma que se diseñe un plan de pruebas y diagnóstico que lleve a saber qué es lo que se espera como resultado. Finalmente es menos costoso y frustrante el regresar y quizá repetir el paso anterior y luego continuar, que construirlo, programarlo y luego volver a diseñarlo.

Sin embargo, estos efectos van mas allá del programador, o el diseñador del sistema, y alcanzan al propio usuario, pues para ellos representa un enorme esfuerzo el haber participado en innumerables entrevistas en busca de información, revisiones, análisis, reuniones de trabajo y quizá un esfuerzo personal y adicional a sus propias asignaciones, lo que en el futuro solo resultará en una tremenda resistencia al cambio, aún y cuando sus sistemas, métodos y procedimientos estén en plena obsolescencia.

Es por ello que se debe observar que existen nuevas metodologías y nuevos caminos que a través de su uso durante todas y cada una de las fases del Diseño y Desarrollo del sistema no únicamente construcción permita validar y probar la viabilidad, coherencia y la exactitud de lo que se ha logrado o desarrollado. El actuar de esta manera nos permitirá detectar errores de definición, de análisis, de necesidad, de construcción, de implantación y de integración, en momentos oportunos en el que el cambiar de curso, táctica o estrategia no resulte costosa y frustrante.

***La importancia de las pruebas de los sistemas radica en que permiten generar productos o sistemas con alta calidad, y a la vez detectar y corregir defectos.***

Las mejores pruebas son aquellas que dan una retroalimentación rápida. Se debe ser capaz de probar tan pronto como se termine un segmento de trabajo. Por esto lo mejor que se puede hacer para prevenir defectos es primero establecer los planes de prueba.

Para realmente entender algo se debe saber como probarlo y viceversa. Esto refuerza el principio de que el probar toma un papel muy importante en prevenir errores, o por lo menos, descubrirlos anticipadamente. "La principal meta al probar no es el encontrar defectos; sino prevenirlos".

El control de las pruebas reduce el riesgo y el costo de hacer funcionar algo que no ha sido correctamente probado. Así cada vez que se habla de sistemas, siempre se debe estar consciente de cuidar las fronteras entre un alto costo contra un bajo riesgo o la relación que existe entre un alto riesgo y un bajo costo.

Este es el punto principal en el que se define el tipo de sistema que se quiere tener, pues es obvio y evidente que un sistema nunca estará completamente probado, y esto por razones de rentabilidad. El punto de equilibrio que permita reducir el riesgo, pero que a la vez permita generar sistemas altamente rentables y confiables, debe ser alcanzado.

Un nivel de pruebas aceptable puede ser determinado y mantenido por un control de calidad. Un sistema está correctamente probado y es funcional cuando se cumple con los requerimientos y especificaciones originales.

Esta es la propuesta principal de esta tesis: el ofrecer técnicas y métodos que permitan diseñar un eficiente y rentable sistema de pruebas para el diseño y construcción de sistemas de información.

Este esquema es planteado considerando ampliamente las directrices definidas en la metodología ISO9000 para el aseguramiento de la calidad, con apoyo directo a ISO9241.

## CAPÍTULO II

### ACTIVIDADES MÍNIMAS DE VERIFICACIÓN Y VALIDACIÓN.

#### Introducción

En este capítulo el objetivo es identificar las actividades de verificación y validación, que durante las fases de vida de un sistema a ser desarrollado, garantizan los resultados que se esperan obtener al crearlo.

Todo sistema de información debe pasar una serie de pruebas y contener ciertas actividades, que con un costo aceptable produzcan el mejor resultado. Es obvio aceptar que, mientras más elementos de control y diagnóstico existan y que mientras mayores sean las pruebas a aplicar al sistema mejores serán los resultados, puesto que se podrán identificar todas las áreas problema y tener la oportunidad de corregirlos. Sin embargo, mientras más se desee probar un sistema mayor será el tiempo que éste tomará para estar en manos del usuario final y de igual manera, el consumo de recursos y el costo serán mayores.

En la mayoría de los casos no se tienen los recursos suficientes para probar un sistema y son tantos y tan variadas las áreas de Oportunidad de posibles fallas, que prácticamente un sistema nunca podría considerarse completa y totalmente probado para evitar cualquier clase de error.

Es por ello que cuando se tiene la idea de desarrollar un sistema, se deben tomar en cuenta algunos marcos de referencia que deben cubrir las respuestas a los siguientes aspectos :

- Presupuesto disponible.
- Utilidad tangible y valor agregado del sistema.
- Beneficios mayores a la inversión.
- Tiempo en que deberá estar disponible.
- Duración de la solución.
- Flexibilidad del sistema.
- Coherencia con los datos corporativos del negocio.

Si se puede identificar sobre qué frontera el sistema deberá ser desarrollado, muchas decisiones que se tomarán en cada etapa serán más acertadas y no se causarán desagradables sorpresas cuando se vaya avanzando en cada una. De igual manera el sistema llegará a ser concluido cubriendo las expectativas que se originaron al inicio del proyecto.

Esto es muy saludable reflexionarlo en su debido momento pues es aquí donde el diseño se inicia, donde el sistema comenzará a cobrar vida, y porque además permitirá evitar el desperdicio de recursos y frustraciones futuras.

Para ello el factor tiempo cobra vital importancia y es muy necesario en esos momentos tener todos los elementos suficientes para la completa elaboración de las pruebas a efectuarse, y es por eso que a lo largo de cada fase del sistema se identificarán las actividades que como mínimas deben ser aplicadas al sistema en cada etapa del ciclo.

## ETAPA DE ANÁLISIS Y DEFINICIÓN

A partir de este momento y hasta la conclusión del sistema, existe un factor que se denomina "riesgo". Y es éste, el que permitirá tomar decisiones respecto al comportamiento de cada función del sistema. Es necesario que el riesgo asociado con el proyecto sea considerado.

Debe recordarse que el riesgo es el potencial de caer o provocar una pérdida, ya sea esta de control, de información o de dinero. Conforme se adicionen más puntos de control y áreas de validación para poder revisarlo, el costo tenderá a elevarse.

Un apropiado nivel de riesgo asociado con un costo aceptable, deberá ser definido previo a las actividades de validación y de verificación.

El análisis de riesgo ayudará a identificar proyectos que la contienen en alta proporción, áreas de posibles defectos y zonas en donde los efectos y defectos pueden ser mayores.

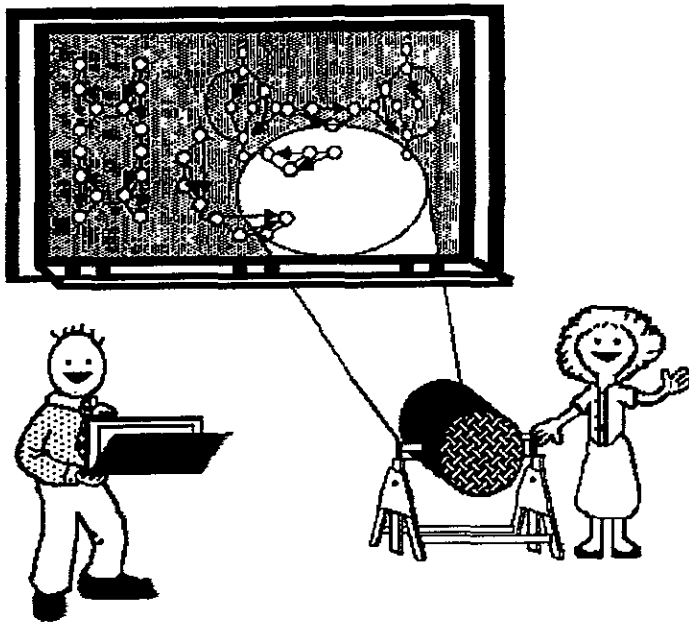
Dentro de un sistema aquellos caminos con alto riesgo son llamados Caminos Críticos, y son considerados de esta manera porque si un error llegara a ocurrir, el sistema entero fallaría; por lo tanto los esfuerzos o actividades de validación y verificación serán enfocados a los caminos críticos y responderán a la siguiente pregunta:

Qué situaciones, circunstancias o actividades provocaron que el sistema fallara en su totalidad ?.

Por ejemplo, considérese un sistema en el cual el 2% de los módulos consumen 60 % del CPU. Estos módulos críticos comprenden un área de alto riesgo para el sistema y deben ser probados efectivamente. Esto se ilustra en la figura II-1

El balance entre las pruebas y costo debe mantenerse, ya que este al concluirse llegará a convivir con el resto de las aplicaciones existentes y a compartir los recursos disponibles, sin degradar la operación.

**FOCOS DE ATENCIÓN EN LOS ANÁLISIS DE RIESGO.**



**"MUEVE LA LUZ HACIA LA IZQUIERDA.  
SI ESA PARTE FALLA, EL SISTEMA ENTERO  
FALLA."**

**FIG. II-1**

## **I.-) Análisis de riesgo**

El análisis de riesgo ayuda a identificar consecuencias no favorables, tales como:

Falla al cumplir los requerimientos:

- ¿Qué consecuencias podrían ocurrir si no se cumple con los requerimientos del sistema?
- ¿Qué consecuencias se podrían presentar si no se cumplen los requerimientos por diferentes áreas del sistema?

Pérdida del control de un proyecto:

- Esto podría ocurrir si no se tiene un sistema adecuado de control de cambios.

El sistema no funciona como había sido esperado:

- ¿Qué sucedería si se cumplen todos los requerimientos establecidos, pero no se satisfacen las expectativas del cliente?
- ¿Qué pasa si futuros proyectos dependen del éxito de éste?

En el análisis de riesgo de un sistema es muy importante evaluar la estructura del proyecto que es la que define cómo está integrado el sistema, cómo es que ciertas entradas de información y parámetros podrán hacer que el sistema en su totalidad pueda integrarse, haciendo el uso más eficiente de todas y cada una de sus rutinas y procesos; ya que no solo es importante tener todos los resultados esperados, sino también la manera en que son obtenidos, pues esto determina la efectividad, flexibilidad y rentabilidad del sistema.

En la evaluación de la estructura de un proyecto es importante considerar la frecuencia de cambios esperados dentro del sistema, ya que es el parámetro que permite identificar el tipo y el número de cambios que razonablemente pueden ocurrir, pues como se comprenderá, no hay una regla que determine el número máximo o mínimo de ellos, aún cuando estos estén documentados y controlados.

Por un lado el efectuar gran cantidad de cambios, aún cuando estos sean pequeños y manejables, traerá consigo un incremento en el costo total del proyecto, pero por el contrario el evitar efectuar los cambios necesarios con la idea de no incrementar el costo del proyecto "ahorrará" dinero y recursos en esta fase, pero estos se pagarán a un excesivo costo cuando el proyecto se encuentre en etapas más avanzadas, y lo que es peor, continuarán aumentando sus costos a medida que estas se desarrollen.

En conclusión los cambios en un sistema deberán ser los menos posibles, puesto que ya se ha efectuado la fase de análisis. Mas sin embargo, no se esta exento de que varias funciones y/o actividades no hayan sido revisadas con la profundidad adecuada y entonces sea necesario aplicar algunos ajustes menores y de ligero impacto a la estructura global del sistema.

Un sistema a desarrollarse puede ser visto de 2 maneras distintas, pero igualmente importantes:

La primera es el análisis individual e independiente, en donde solo interesa la parte interna y propia del sistema, pero en el caso de actividades mínimas de verificación y validación lo que reviste mayor importancia es el segundo análisis: el grupal, el de integración y el de convivencia.

Aquí la importancia del análisis y diseño radica en identificar las áreas de mayor problema cuando el sistema se encuentre en un ambiente productivo y conviviendo con el resto de las aplicaciones.

Aspectos altamente relevantes deberán considerarse, como por ejemplo:

- ¿Cómo el sistema consume recursos?
- ¿Son aceptables y suficientes?
- ¿Cómo se combinan los datos: en forma natural, o deben crearse numerosas interfaces y conservarse archivos con imágenes distintas?
- ¿Cómo se presenta el control de la integridad referencial?
- ¿Cómo se asegura la alimentación y actualización de una fuente única de datos?
- ¿Qué funciones de control se adecuarán o se eliminarán?
- ¿Qué áreas participan ahora y cuáles dejan de hacerlo?
- ¿Qué puntos de control son nuevos y como se manejan?
- ¿Qué seguridad y vigilancia requieren nuestras nuevas aplicaciones?

Estos cuestionamiento se deben aplicar y contestar antes de iniciar la construcción del nuevo sistema. Y es ahí precisamente donde los marcos de referencia se confirman o se rechazan; donde la teoría se vuelve práctica.

Este análisis del impacto sobre las nuevas y actuales aplicaciones, es vital para determinar la vida del sistema, así como su viabilidad, pues es obvio que si no se tiene los recursos suficientes el sistema demandará obtenerlos e invertir en ello, o ir recortando las expectativas del mismo.

Cualquiera que sea la decisión se debe volver a validar el factor riesgo, pues es factible que el sistema exija inversiones tan grandes, que deje de ser rentable, o que deba ajustarse a tal grado que se vuelva no útil si muchas de las funciones quedan solo en ideas.

Quando se habla de crear nuevas aplicaciones, generalmente existe un espíritu de modernidad y renovación, y conforme se analizan las funciones que se deben cubrir, se cae cualquiera de 3 aspectos típicos:

- El primero es cuando los recursos y tecnología usadas hasta ahora son suficientes y se pueden desarrollar aplicaciones que aseguren resultados al menos durante 5 años. Así que solo es necesario y suficiente una excelente documentación del uso, operación y mantenimiento de nuestro nuevo sistema, pues gran parte de las herramientas tecnológicas que se usarán ya son conocidas por el usuario final y también por los programadores.
-



Aquí ya existe algún grado de experiencia en el ambiente actual y solo debe cubrirse el mantener, tanto de parte del usuario como de quien da el mantenimiento, que exista siempre un respaldo o alguien que en cualquier momento pueda substituir a quien normalmente trabaja con el sistema. Es sorprendente ver cómo sistemas que fueron correctamente planeados, diseñados y administrados, comienzan a perder su esencia y filosofía porque quien los usa o los atiende, no los conoce lo suficiente, y se empiezan a crear interfaces no necesarias, modificaciones y ajustes que resuelven los problemas de primera instancia, pero que no conservan sus principios básicos.

Se debe aceptar que son las necesidades y consecuentemente las aplicaciones las que llevan a cambios tecnológicos, pues el Hardware debe responder a las demandas del Software: mayor rapidez, más capacidad, mejores resultados, menores costos.

El software debe también actualizarse y ajustarse a sus propias demandas, sin embargo, es un hecho que este no ha evolucionado con la velocidad con la que lo ha hecho el Hardware, y es por ello que todos los elementos que se tengan al alcance deben utilizarse para crear un mejor desarrollo de aplicaciones.

Resultado de esta búsqueda aparentemente sin final, es que la gente del medio informático debe tener acceso a una capacitación constante, moderna y eficiente, en que las nuevas vanguardias se combinen con los costos y las nuevas necesidades de información.

- El segundo punto es cuando al querer desarrollar alguna aplicación, es necesario el recurrir a nuevas herramientas que no modifican el ambiente actual, que complementan y adicionan una variedad de aplicaciones. Nuevamente el conocer el software y las herramientas técnicas se hace indispensable.
- El tercero y último punto es cuando se utilizan nuevas herramientas de desarrollo, nuevo equipo y las relaciones anteriores ya no funcionan igual. El cambio es drástico e impactante.

Entonces es necesario apoyarse en los siguientes principios :

- Entrenamiento
- Sistema de documentación de alta calidad
- Mantener una estrecha comunicación y convivencia con el usuario final. Si es posible permanecer físicamente en las mismas instalaciones y ser prácticamente compañeros.
- Mantener un estricto sistema de control de cambios.
- Efectiva comunicación, real y constante.
- Estabilidad en la organización cliente/usuario.

Evaluar en que situación se encuentra el sistema que se desea desarrollar permitirá asegurar que se han considerado los costos, beneficios y la factibilidad de su instrumentación, ya que no es suficiente saber qué rutinas, qué programas, reportes e información debe recibir y producir el sistema, pues esto no únicamente determina su tiempo de vida y por ende su éxito.

En los aspectos técnicos se debe efectuar un profundo análisis del tipo de equipo y el soporte que existe para los programas producto.

En incontables ocasiones se adquiere Software que es muy eficiente y que promete una alta productividad, pero que cuando se analiza el mercado de servicio, se encuentra con que nadie lo conoce, que los técnicos especializados se encuentran en el extranjero y que las limitantes entre los tratados de transferencia tecnológica, son tan altas que resulta muy costoso el mantenerlo, capacitar al personal, adicionar o combinar aplicaciones, o recibir el entrenamiento para el usuario; y quizá se deba retornar a alguna otra opción, que en un principio no fue completamente analizada. Esto únicamente traerá frustraciones y grandes pérdidas económicas.

Debe entenderse que una nueva aplicación no es solamente más programas, archivos o datos, sino que es todo un proceso hacia la solución de problemas que deben ser de raíz y en forma definitiva.

Otros puntos de análisis muy importantes son:

- **Las interfaces con sistemas externos y su crecimiento.**

Una vez que se ha analizando cada aspecto, se debe ir registrando todo cuanto se observa, a fin de poder obtener una panorámica real, un diseño concreto y un resultado contundente que defina la factibilidad del sistema. Esto añadido con el siguiente punto, son la mejor técnica que mostrará qué tan real y rentable puede ser un sistema. Y hasta ahora solo se ha invertido en actividades de análisis. En este punto cambiar de opción, estrategia o proseguir, siempre resulta más exitoso y económico que ahorrar tiempo y continuar hasta que algún problema o limitante técnico, educacional o de capacidad detenga el desarrollo, cuando pudo haberse evitado con una increíble anticipación y a un costo realmente mínimo.

- **Tamaño del proyecto:**

El poder dimensionar el alcance del proyecto permitirá identificar su factibilidad y posibilidad de logro.

La creación de un plan de acciones y actividades mayores así como un análisis punto por punto de la función de cada parte del sistema dará fácil y rápidamente la dimensión real del proyecto, hará recapitular algunos pormenores y validar nuevamente las ideas originales.

Se debe considerar que esto es un método de estimación del tamaño de un proyecto usando su funcionalidad y no cuantas líneas de código o programas se requieren.

Hasta este momento ya se tiene suficiente información, a un costo muy bajo para tomar decisiones respecto a que se ajusta, se cancela o que se mantiene de la propuesta y objetivo inicial.

Como resumen de los puntos anteriores se puede concluir que los factores de riesgo son ajustados y determinados por:

- El establecimiento de los objetivos de las pruebas; es decir, saber qué se quiere lograr y encontrar.
- La definición del procedimiento de control.

## **II.-) CONTROL DE CAMBIOS**

A través de las experiencias se ha podido aprender a vivir con un principio básico: "No hay nada más permanente ni más constante que el cambio", y esto es especialmente cierto en el desarrollo de sistemas. Esto es debido a que las necesidades no son estáticas sino por el contrario son tremendamente dinámicas, ya que se modifican día a día, a medida que los intercambios comerciales surgen, las economías evolucionan, las sociedades cambian y progresan, en que las condiciones geográficas y las distancias se acortan. Todo permanece en movimiento y las respuestas deben también cambiar y evolucionar.

Es obvio que un procesador que hace 20 años dio respuesta a una cierta cantidad de necesidades, ahora no lo puede hacer, y simplemente porque esas necesidades ya no son las mismas, han crecido en volumen y en características.

Los procedimientos de control de cambios deben estar diseñados para controlar y asegurar que si un cambio es aprobado, debe registrarse y aplicarse en todas las salidas que estén relacionadas, aunque sea mínimamente. Para establecer un eficiente control de cambios, se sugieren las siguientes actividades mínimas:

### **II.A.-) Identificar las líneas básicas.**

Las líneas básicas son los productos del ciclo de vida de los sistemas, que lo describen durante las etapas de desarrollo. Hay típicamente 3 líneas:

- Especificaciones de requerimientos
- Especificaciones de diseño
- Implantación del producto

Como regla general, el 98% de las salidas de cada fase del sistema caen en estas 3 líneas básicas.

---

## **II.B.-) Procedimientos para establecer requerimientos formales de cambio.**

Los procedimientos de cambio definen la manera en que estos se controlan. Estos pueden ser formas prediseñadas o un sistema interactivo.

## **II.C.-) Actividades de Aprobación.**

Determinar quién tiene la autoridad para aprobar los cambios. La autoridad puede tenerla un usuario, un analista o un grupo de ellos, pero debe observarse que deben existir diferentes niveles de cambio, que requieran de diferente nivel de aprobación. La gerencia de sistemas y el usuario deben estar involucrados en la decisión de hacer los cambios.

## **II.D.-) Impacto del cambio requerido.**

El procedimiento elegido para el control de cambios necesita incluir un método para determinar el efecto de un cambio en el sistema.

Si no se considera que cada cambio es importante, ya que por pequeño que este sea tendrá un impacto sobre el sistema, pronto se encontrará rodeado de muchas notas que dirán "tener cuidado con ciertos aspectos", pero que en su conjunto causan un impacto de importancia bastante considerable. Sin embargo, hay cambios que podrán parecer sencillos, y quizá triviales, pero que su constante aplicación, o el resultado que generan, es tan utilizado por varios procesos, que a pesar que inicialmente se *ocultaban como algo sencillo*, resultan ser de grandes dimensiones.

Todo cambio requerido debe documentarse hasta el más mínimo detalle. Se debe obtener el mayor nivel de información posible resultado de un análisis, que indique: quién generó el cambio, en qué consiste, cuánto tiempo estará vigente, cómo se integra a la arquitectura del sistema, a qué entradas y salidas afecta, cómo será alimentado, restricciones y condiciones especiales, tipo de complejidad, mantenimiento posible, y finalmente dimensión y urgencia para aplicarse, así como sus consecuencias y beneficios de implantarlo o no.

Cuando un cambio es analizado no debe perderse de vista la vital importancia que representa el estimar cuánto tiempo llevará aplicar el cambio y cuánto afectará al plan de trabajo. Esto reviste especial interés cuando más grupos de trabajo son afectados, pues *es muy posible que estén desarrollando actividades en forma paralela*.

Cuando los cambios no son debidamente documentados y justificados, en algún momento del sistema se verá que alguien no está de acuerdo con ellos; y se hará necesario regresar a investigar y obtener información que en momento pudo ser de mayor utilidad. Esto generará conflictos, pérdida de tiempo y recursos, así como un evidente retraso en el proyecto.

## **II.E.-) Implantación del cambio.**

Una vez que un cambio ha sido analizado, documentado y aceptado, lo que procede es implantarlo. Y en este punto todos los módulos, procesos, diseños y productos afectados, necesitan ser probados nuevamente utilizando los mismos casos de prueba diseñados originalmente.

Los nuevos resultados deben ser comparados con los anteriores para garantizar que los cambios no generaron nuevos errores. Esta técnica se denomina **PRUEBA DE REGRESIÓN**.

Cuando se desarrollan pruebas de regresión todas las diferencias entre los resultados anteriores y los nuevos deben ser atribuidas a cambios intencionales.

A lo largo de este capítulo se ha podido constatar que sin importar lo que cambie, se mantenga o se deba considerar para el proyecto, todo debe de estar debidamente documentado.

Pero paradójicamente, todo mundo coincide en que mientras mejor y mayor información se tenga mejores serán los resultados. Más aún todo mundo exige documentación y está de acuerdo en su importancia, sin embargo, es el punto más débil del área de Desarrollo de Sistemas: una completa y actualizada información.

Por lo anterior se recomienda generar la documentación conforme el tema avanza.

## **III.-) DESARROLLO DE PLANES DE PRUEBA**

Este punto es quizá uno en los que mayor tiempo se deberá invertir, pues a diferencia de lo que durante años se hecho, no solo se deben diseñar planes de prueba en el momento de llegar a la fase de construcción o de prueba del programa necesario. Sino por el contrario, los planes de prueba deben incluir todas y cada una de las fases que componen la vida del sistema.

Los planes de prueba deben ser creados lo antes posible, pues a través de ellos se describe el alcance del esfuerzo de pruebas que el proyecto va a contener. Se identifican los objetivos mayores y críticos, se analizan los pruebas unitarias y de sistema, así como también las pruebas que determinarán la aprobación del usuario.

No existe, ni es objetivo de esta tesis, el definir formas o mecanismos únicos para el desarrollo de los planes de prueba, pues estos estarán definidos por el tipo, tamaño y naturaleza del proyecto, y deben indicar a que se quiere llegar y como saber que se ha conseguido.

---

### **III.A.-) Objetivos de los planes de prueba.**

Los planes de prueba deben describir el calendario y los recursos requeridos para cada actividad de prueba en todas las fases del sistema. Estos al igual que los planes de aceptación del usuario, son creados en las fases de Definición y Análisis, porque es cuando se requiere que la información esté disponible.

Los planes para la generación de pruebas integrales del sistema se crean durante la fase de Diseño, y los planes de unitarios son creados y ejecutados durante la fase de Construcción. Los objetivos que debe cumplir un buen plan de pruebas son:

- Especificar el objetivo de cada prueba, el método, procedimiento y técnica con el que este se alcanzará.
- Especificar el camino crítico y el alcance que se cubrirá con las pruebas.
- Identificar los recursos requeridos para realizar las pruebas para que estén disponibles cuando se requieran.
- Descubrir y corregir los posibles productos antes de que lleguen a convertirse en defectos.
- Identificar responsables de cada prueba y las fechas en que se deben concluir.

La mayor ventaja de definir estos planes, es que el hecho de desarrollarlos es una excelente técnica de prevención de errores y defectos y además cuando el sistema esté listo para ser probado se tendrán los recursos y tiempo para realizarlo en forma planeada y organizada.

### **III.B.-) Información contenida en los planes de prueba.**

Cada plan de prueba deber ser identificado particularmente, pues esto hace más fácil reconocerlo.

- Alcance: Listar las partes que serian probadas en este plan, por su identificador único y número de versión, así mismo documentar los objetivos de probar esta parte del sistema.
  - El objetivo de las pruebas del sistema es determinar si su desempeño esta dentro del límite de tolerancia y demostrar su funcionalidad al usuario.
  - Recursos requeridos: Describir el software y hardware requerido para conducir las pruebas. Listar el personal de soporte y las herramientas que serán necesarias y por cuánto tiempo.
  - Calendario, plan de trabajo y Programa: Es necesario efectuar estas 3 programaciones a fin de estimar el tiempo necesario para poder aplicar las pruebas y reconocer el momento de hacerlas, y así tener la oportunidad de corregir los defectos y volver a probar antes de liberarlo al cliente.
-

- **Criterios de validación:** Cuando se aplican pruebas, éstas deben estar bajo ciertos criterios que den un resultado prácticamente binario: es correcto o no lo es. Y para ello durante todas las pruebas, estos criterios deben ser consistentes y cualquier persona involucrada en el proyecto debe poder aplicarlos.

Estos criterios deben de ser definidos conjuntamente con el usuario final.

Posterior a este punto, algo que no debe omitirse por ningún motivo, es el de administrar la documentación que se genera. Es prudente que cada participante del grupo de desarrollo y administración del proyecto, la lea frecuentemente y con detenimiento. Esto genera un autoanálisis y un buen nivel de autocritica que podría resultar en aspectos muy positivos. Por ejemplo: determinar en dónde se están creando los mayores errores e inconsistencias, y datos que muestren en donde se degrada el funcionamiento correcto del sistema.

Aspectos generales, pero de gran ayuda, son los que se listan a continuación y que pueden dictar marcos de referencia para que el usuario y el analista de sistemas, definan conjuntamente los criterios de validación para el sistema:

- La compatibilidad con otros sistemas
- La seguridad mínima y optima necesaria y niveles del mismo.
- Puntos de recuperación y efectividad para cada tipo de información.
- Compatibilidad con varias configuraciones de software y hardware.
- Niveles de información, guías y ayuda, validación y control de errores.

Al aplicar los criterios de validación se deben incluir revisiones relacionadas con pruebas funcionales del camino crítico y de interfase con el cliente y pruebas de configuración y desempeño, que deben cubrir volumen, capacidad y tiempo de respuesta.

### **III.C.-) Plan de pruebas de aceptación del usuario**

El plan de pruebas de aceptación del usuario debe hacerse durante las 3 primeras fases (planeación de la información, definición del problema y análisis), para de antemano estar preparado y reforzar el conocimiento de lo que el sistema debe lograr.

El ir preparando y desarrollar estos planes de prueba, evitará improvisar y generar el descontrol que pudiera darse al momento de revisar el sistema con el usuario, quien a su vez tiene sus propios puntos de vista de cómo verificar la correcta operación del sistema.

Las pruebas de usuario son del mismo tipo que las que se ejecutaron durante el periodo de prueba del sistema. La principal diferencia entre estas es el objetivo que cada una persigue.

El objetivo de las pruebas de sistema es descubrir defectos. El objetivo de las pruebas de aceptación del usuario es demostrar que los criterios, previa y claramente definidos, se han cubierto.

***“NO QUEREMOS DESCUBRIR ERRORES DURANTE LAS PRUEBAS DE ACEPTACIÓN DEL USUARIO”.***

Una estrategia muy simple, pero que reduce sorpresas y dudas, es el ejecutar previamente y sin presencia del usuario, cada una de estas pruebas, para asegurar que funcionan sin falla ante él.

Al momento de aplicar las pruebas de aceptación, no importa quien las conduzca, el usuario o el analista; lo importante es que no se omita ningún aspecto y que los criterios y especificaciones sean mutuamente acordados. En algunos casos el usuario insistirá en aplicar sus propios planes de prueba, analizar al azar algunas funciones; lo que es probablemente inevitable y debe permitírsele hacerlo. Esto le agregará mayor satisfacción y confianza.

Toda persona que participe en estas pruebas debe estar preparada para invertir mucho tiempo a explicar los resultados obtenidos en cada prueba.

Si a pesar de todo lo que se ha mencionando: se aplicó, desarrolló y se documentó adecuadamente, se encuentran errores en este período, esto puede deberse a dos razones:

- Un mal entendido en las especificaciones
- Que el sistema se comportó adecuadamente ante un caso de prueba mal definido o con entradas de información erróneas.

Sin embargo, si los planes se llevaron a cabo correctamente, cualquier error que surgiera será muy fácil de corregir.

Como puede verse claramente en todo este proceso, el cliente final tiene una alta participación y los analistas de sistemas y el líder del proyecto, deben asegurar que así sea, pues si el cliente no está interesado en el desarrollo del sistema y su evolución, nunca estará interesado en probarlo o usarlo.

Durante las fases de Definición y Análisis las pruebas deben enfocar los esfuerzos a organizar la información. El análisis de riesgo debe ser ejecutado para determinar el nivel apropiado y que costo sea aceptable, ya que este puede ayudar a identificar los módulos más críticos en el sistema total.



De acuerdo con estas 2 fases se efectuarán las siguientes funciones:

- Registro y reporte de problemas
- Análisis de defectos y errores
- Control de cambios
- Planes de Prueba de Aceptación del Usuario
- Planes de Prueba del Sistema.

De esta manera se podrá garantizar que al aplicar estas actividades consideradas como mínimas para verificar y validar el éxito y buen progreso de un sistema, se podrán obtener resultados aceptables que den vida y continuidad al proyecto, el cual estará orientado de la mejor forma a cumplir los objetivos para los que fue creado.

## **FASE DEL DISEÑO DE NEGOCIOS Y DISEÑO TÉCNICO**

En gran parte de las metodologías existentes, esta fase no se encuentra definida en forma separada, pero hoy por hoy, los recursos que están disponibles y en los que se habrá de invertir, deben ser cuidados de tal forma que se eviten desperdicios de cualquiera de ellos, en términos de dinero y tiempo.

Con esta fase se deberán completar y conjugar los análisis resultantes de la información obtenida del cliente de las funciones que se desean cubrir y de la definición del problema; en fin de todas y cada una de las fases previas. Y es aquí que toda esa información que se encuentra por grupos o por áreas comenzará a tomar una forma real y casi física, ya que estará integrando cada interfase y función que el sistema deberá ejecutar, sin necesidad de llegar hasta la construcción del sistema mismo para revisar y probar que eso es lo que se quería obtener.

La posibilidad de que este diseño cumpla con las expectativas y objetivos planeados, será increíblemente alta, si es que se aplican las técnicas de prueba previas.

De cualquier manera al concluir estas fases, los siguientes aspectos deberán tener respuestas y fundamentos acordes a los análisis, planes y pruebas previas, y a los objetivos establecidos:

- Satisfacción de los requerimientos originales.
- El diseño es resultado de gran participación del usuario.
- Se tienen las facilidades para la implantación.
- Es real y factible.
- Se proporcionó la documentación al cliente para cuestionar la operación.
- La arquitectura tiene capacidad para el crecimiento.
- El diseño físico de los datos cubrió las necesidades de volumen de datos, tiempo de respuesta y frecuencia de acceso.

### **I.-) Planeación de las pruebas de integración.**

El objetivo de las pruebas de integración es descubrir defectos en las interfaces y el intercambio de datos, entre varios modelos y subsistemas que lo componen. La información necesaria para la planeación de las pruebas de Integración viene de los diagramas de la estructura del sistema, del modelo de datos y de proceso, y estos definen la estrategia de integración y las acciones para esta.

La estrategia de integración definida en el plan de pruebas, determinará el orden en el cual los módulos deberán ser codificados y probados. Los módulos deberán ser desarrollados al mismo tiempo y orden en el que requieren ser integrados.

Este proceso descubrirá muchos defectos que deben corregirse antes de realizar cualquier prueba. El software debe ejecutarse antes de que se inicie cualquier prueba seria de integración.

---

### **I.A.-) Big Bang.**

Una estrategia de integración es la llamada Big-Bang o Fuerte Estallido. Su principio básico es la integración total del sistema y después probarlo.

Esta técnica está caracterizada por una rápida integración: Todos los módulos son integrados al mismo tiempo y es por ello que se le llama "fuerte estallido" pues todos los posibles defectos y anomalías saldrán juntos.

### **I.B.-) Pruebas de integración incremental.**

La Integración incremental del sistema tiene 2 variedades: "Top Down " (arriba/abajo) y "Bottom Up" (de abajo hacia arriba). Estas estrategias no deben confundirse con el diseño de sistemas Top Down y Bottom Up; no tienen relación ni en forma ni contexto.

La integración incremental puede ser ejecutada en conjunto con las pruebas unitarias o modulares. Un módulo que no ha sido probado, puede estarlo con los módulos que lo invocan y que ya están probados. Esto facilita la tarea de las pruebas modulares debido a que los módulos de prueba no necesitan crearse en ese momento.

- **ARRIBA/ABAJO**

En esta técnica, los módulos se integran iniciando de arriba y se trabaja un nivel hacia abajo al mismo tiempo.

- **ABAJO/ARRIBA.**

Los módulos a diferencia de la técnica anterior, se integran iniciando de abajo y trabajando hacia arriba.

De cualquier manera, ya sea aplicando la técnica de BIG BANG, TOP DOWN o BOTTOM UP, al final de cada una de ellas, se debe proceder a lo que se denominará pruebas de conformidad.

Las revisiones de conformidad deberán cubrir:

- Hardware, teleproceso, requerimientos de software.
  - Modelos del sistema
  - Reportes, formas y pantallas
  - Definición de los elementos de datos
  - Especificación de procesos
  - Diseño de documentos
-

- Especificaciones para la guía del usuario
- Programas de entrenamiento
- Modelos de Organización
- Especificación de programas
- Calendario maestro del sistema
- Definición física de los datos
- Flujo del sistema
- Estructura del sistema
- Especificaciones de la guía de operación
- Plan de conversión.

Cada uno de estos aspectos ayudará significativamente a garantizar el éxito del sistema. Lo importante será realizar un breve análisis que ayude a determinar qué técnicas de prueba integral se aplicarán al sistema en desarrollo. Para ello es muy importante evaluar las ventajas que cada una ofrece y tener en cuenta el tipo de sistema y aplicación que se tiene y hacer un comparativo para obtener la máxima relación costo-beneficio.

A continuación se describen las ventajas y desventajas de cada una de las técnicas, a fin de apoyar la decisión sobre cuál debe ser utilizada.

### **Integración de Bing-Bang (Fuerte estallido)**

#### **Ventajas:**

- La codificación de unidades y sus pruebas pueden hacerse en paralelo porque estos no tienen dependencias.
- El sistema estará disponible y funcionando mucho antes, que usando cualquier otra técnica.

#### **Desventajas:**

- Las rutinas o módulos que llaman a la unidad que será probada para proporcionable datos de entrada y aquellas rutinas que simulan la acción de módulos, que serán llamados por la unidad a ser probada, deberán ser creadas para poder ejecutar la prueba.
- Debido a que estas rutinas van a ser creadas para todos los módulos, las pruebas tomarán mayor tiempo y los defectos permanecerán más tiempo en el sistema.
- La revisión e identificación de defectos será más difícil, debido a que hay más lugares que analizar.

### **Integración Top-Down (arriba hacia abajo).**

Esta técnica es independiente de la manera en que fue diseñado el sistema. Una es la manera en que se diseña el sistema y otra la manera en que se probará en forma integral.

El nivel más alto se prueba primero, y los módulos subsecuentes quedan en espera de ser probados. Después de que el primer nivel ha sido individualmente probado, se continua con el siguiente nivel, y al concluir se integran y se continúa con el nivel siguiente, se vuelve a integrar, y así sucesivamente hasta concluir el sistema total. Esto queda ejemplificado con la figura II-2

#### **Ventajas:**

- El sistema estará disponible antes que usando la técnica de Bottom Up.
- Es muy bueno para propósitos de demostración y para resolver errores humanos. El sistema solo tendrá limitaciones funcionales porque gran parte del sistema está simulado con rutinas de salida.
- Una vez que las funciones de entrada/salida están en su lugar, las pruebas subsecuentes serán más fácil de desarrollar.
- Las rutinas de entrada o que llaman a la unidad a ser probada, no es necesario desarrollarlas.

#### **Desventajas:**

- Puede haber la tentación de combinar la integración Top-Down con el proceso de diseño, en otras palabras, iniciar la codificación y las pruebas de los módulos de más alto nivel, antes de que los módulos de bajos niveles se hayan diseñado.
- El no resistir la tentación de diseñar los módulos de menor nivel, podría causar cambios de diseño, en los niveles altos. Si ya se tienen codificados y probados e integrados los módulos de más alto nivel, hacer un cambio a los mismos puede ser costoso y traumático.
- El diseño detallado de todos los módulos debe ser completado antes de codificar para lograr minimizar el retrabajo por diseños incompletos.

Después de que B, C, y D han sido probados individualmente, son integrados y probados. Las rutinas de salida son usadas en lugar de los módulos E, F, G, H e I.

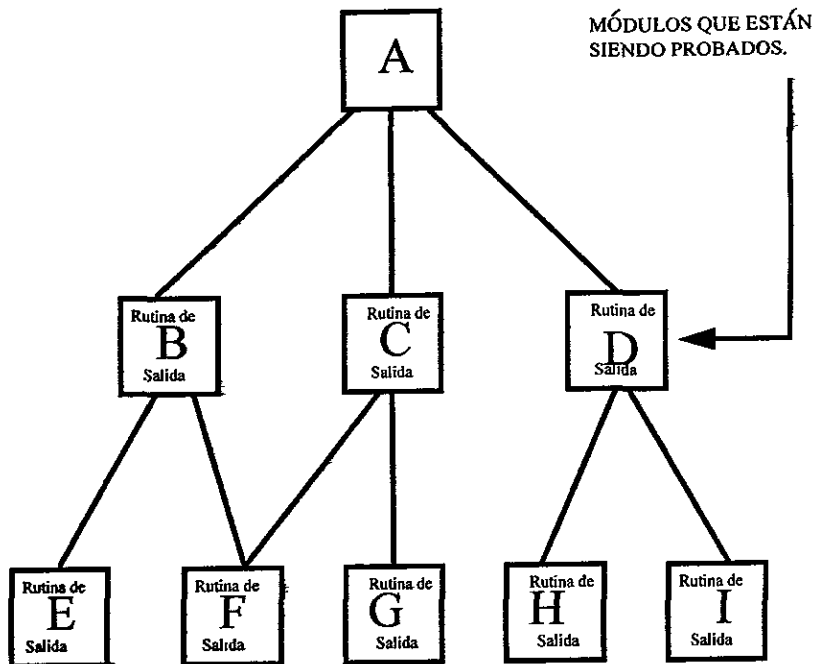


FIG. II-2

### **Integración Bottom-Up (de abajo hacia arriba).**

Esta estrategia de integración es muy similar a la de Top-Down, pero en forma inversa.

#### **Ventajas:**

- El módulo en cuestión es directamente probado.
- Las entradas son más fáciles de crear y las salidas más fáciles de observar.
- Es posible tener mayor flexibilidad en el calendario y en el desarrollo del paralelo.
- Las rutinas que serán invocadas por este módulo no requieren ser desarrolladas previamente.

#### **Desventajas:**

- El sistema no puede verse integrado en forma breve, se requiere de mucho tiempo para lograrlo.
- Los problemas del medio ambiente en el que se prueba o de las interfaces externas no se hacen evidentes sino hasta muy tarde.

***La mejor estrategia de integración incluye elementos de cada una de las tres estrategias descritas anteriormente:***

- 1) Los fundamentos de Big Bang.***
- 2) Los controles de Top-Down.***
- 3) Todo Bottom-Up.***

Los elementos fundamentales del sistema (funciones de entrada/salida, manejo de memoria, y controles simples), deberían ser una unidad probada e integrada toda en una sola. Esto proporcionará estructura y visibilidad al sistema e identificará los problemas del ambiente y de las interfaces externas.

Las estructuras de control complicadas pueden ser integradas usando TOP DOWN para hacerlo con limitantes (uno o dos niveles abajo solamente).

Usando esta estrategia:

- El sistema se tiene estructurado y visible.
- Los problemas de control se resuelven con anticipación.
- Se reduce el uso de rutinas de salida complicadas.
- Se identifican rápidamente los problemas de manejo de recursos.

Y con esto se dan por concluidas las revisiones principales.

Reviste gran importancia el ejecutar las pruebas de cada módulo con el equipo de trabajo y el usuario. Pues es aquí el punto crítico de toda la vida del sistema.

***Se concluye que hasta aquí solo podrían incluirse algunos cambios menores y de bajo costo, y de ser así será un buen indicativo de que todo el trabajo ha sido realmente bueno.***

Pero si por el contrario, se identifican grandes lagunas, muchas preguntas sin respuesta y cambios fuertes, es mejor detenerse y regresar hasta el principio, ya que aunque resulte costoso y frustrante, será mucho más productivo y económico que continuar, pues evidentemente hacerlo causará problemas, y es mejor confrontarlos en esta fase que llevarlos al fracaso rotundo.



## FASE DE CONSTRUCCIÓN.

Al llegar a la fase de construcción del sistema, prácticamente se tiene una seguridad del 100% de que el diseño esta completamente validado, probado y aceptado; que cada uno de los análisis tienen el nivel de profundidad y detalle necesario para poder implantar el sistema, y existen tanto los planes de prueba de conformidad y de globalización del sistema, como la secuencia en la que cada módulo será probado.

En esta fase son desarrollados los planes de prueba unitaria con gran detalle de pseudo-código. Así que los programas definidos como unidades son codificados y probados, además, aquellos componentes que no son programas deben ser también probados.

Una unidad de prueba es uno o más módulos de un programa en conjunto, con cualquier control de datos y procedimientos de operación.

Por ejemplo, una unidad de prueba podría consistir de subrutinas FORTRAN, PL/I o quizás rutinas de COBOL.

De cualquier manera, las unidades de prueba deben seleccionarse para ser convenientemente probadas; y para ello deben contener los siguientes atributos:

- Todos los módulos deben pertenecer a un mismo programa.
- Al menos debe existir un módulo sin haberse probado.

Estos atributos garantizan que exista alguna razón para ejecutar la prueba y que esta no sea redundante. La estrategia de integración que se haya planeado será de gran influencia en la manera en que se crean las unidades de prueba.

- La unidad de prueba debe ejecutar al menos una función completa.
- La unidad es el único objetivo de la prueba.

Quizá en la fase de construcción se pueda encontrar que una unidad de prueba podría aplicar para varias o todas las unidades en el sistema, pero sin embargo, y para lograr objetividad, cada unidad deber ser probada en forma separada e independiente.

La prueba de la unidad debe ser el trabajo del analista. El ingeniero de sistemas o el analista debe probar y depurar la unidad por él mismo.

La unidad debe tener la documentación de su especificación original. El analista requiere tener las especificaciones de la unidad bien definidas, para poder apoyarse en los casos de prueba cuando estos deban ser desarrollados.

La unidad es un producto visible e identificable.

Esta debe ser capaz de ser compilada y/o ensamblada y probada en forma independiente y finalmente integrada al sistema, ya que de esta manera cada función tendrá un inicio, un proceso y un fin.

### **Objetivo de las pruebas unitarias**

El objetivo de las pruebas unitarias, al igual que cualquier prueba, es comparar las funciones actuales de cada unidad contra las funciones esperadas y definidas en los requerimientos. Esto se hace para encontrar los defectos que hasta ese punto no se hayan descubierto.

Las pruebas unitarias es el proceso donde el sistema en desarrollo pasa de un ambiente privado a un ambiente de dominio público. Pues el coordinarse para conservar los datos, casos prueba, planes de prueba y secuencia, propicia un fuerte intercambio de opiniones, ideas y experiencias entre los participantes.

El hecho de que se recomiende definir Unidades de prueba a partir del conjunto total del sistema, es por razones principalmente de control de la misma y de sus resultados, así como de las acciones a seguir posteriormente. El hacerlo por unidades de prueba, que completen un pequeño ciclo y definan una función particular, permitirá efectuar pruebas simples, ya que son más pequeñas y manejables.

El ejecutar un análisis de causa y efecto sobre una función en particular resulta más fácil, manejable y objetivo que sobre todas las funciones o varias a la vez. Como resultado de ello, las correcciones son fácilmente identificables y su aplicación será considerablemente sencilla.

Al definir unidades de prueba, se obtiene una gran flexibilidad en la secuencia a seguir para probar, y estas pueden ser probadas antes de que todo el sistema este concluido, o muchas pueden probarse en forma simultánea. Y de esta manera se logrará una integración gradual antes de que las unidades hayan sido concluidas.

### **Desarrollo de una plan de prueba de cada unidad**

En esta fase de construcción del sistema, la primera actividad que debe efectuarse es el desarrollo de un plan de prueba para cada unidad en el que se describa claramente cómo es que se llevará a cabo el proceso de prueba.

Cada plan de prueba incluye:

- Alcance.
- Técnicas a utilizar.
- Recursos requeridos.
- Calendario de actividades.

Sin embargo, no se hace necesario que se escriba un plan de prueba para cada unidad seleccionada, debido a que uno podría ser aplicado a varias unidades cuyas funciones estén muy identificadas y tengan estrecha relación.

El objetivo de cada plan está enfocado al nivel de cobertura deseado y la validación de los criterios que la unidad debe cumplir, y en ello participan las especificaciones de los programas, el modelo de relaciones de cada elemento, las bibliotecas de programas y los casos de prueba. Incluir la existencia de problemas en el calendario de actividades original y el costo de cada una, permitirá siempre estar en un punto muy objetivo de lo que se probará.

Toda prueba, aún la más mínima, consume tiempo y recursos, que se irán restando del total disponible para asegurar que el sistema funciona correctamente. Por tal motivo se debe tener identificado lo que es estrictamente necesario probar en forma exhaustiva, parcialmente o ideal.

Algunas veces no podrá probarse todo lo que se quisiera, por ejemplo: el plan no permitirá la ejecución de 256 variaciones de colores en una pantalla, pero si puede permitir hacer pruebas críticas para ver que la pantalla puede ser usada por daltónicos.

Como puede verse los planes son los que ayudarán a limitar las pruebas. El punto interesante aquí es el asegurar que se ha tomado una decisión clara y bien fundamentada acerca de qué pruebas no ejecutar, en vez de ejecutar pruebas y más pruebas hasta que el tiempo y el dinero se acaben.

Es necesario al limitar las pruebas el considerar los siguientes aspectos:

- Evitar pruebas innecesarias de proveedores de Software/Hardware
- Eliminar pruebas no esenciales (errores de I/O del hardware, overflows y otros eventos poco usuales).
- Enfocarse a las partes críticas del sistema, por ejemplo: probar caminos de I/O y adjuntos, en vez de los colores de la pantalla.
- Probar rutinas comunes primero.
- Usar un diseño modular
- La decisión de no probar un camino o función de una unidad tiene que ser revisada y aprobada en forma grupal por los responsables de las pruebas.
- Si una unidad no se prueba dinámicamente debe asegurarse de que el código sea inspeccionado durante la codificación.

Todos los casos de prueba requeridos para lograr los objetivos de las pruebas deben ser desarrollados. Si después se decide que esos casos no serán ejecutados, existe un documento que soporta la razón de la excepción.

Cuando se inicia la tarea de querer probar una unidad del sistema, frecuentemente se trata de demostrar que lo que se hizo funciona. Sin embargo, esta actitud no es muy objetiva y oculta errores. Es por ello que al probar una unidad, la estrategia y actitud que debe asumirse es la de encontrar el mayor número de errores, es decir; las pruebas unitarias deben continuar hasta que los objetivos sean alcanzados.

Cuando errores o defectos son detectados deben ser analizados y corregidos de la manera más adecuada, que mantenga la filosofía del sistema, sin permitir hacer solo ajustes o malas adecuaciones con tal de solucionarlo rápidamente. Posteriormente, los casos de prueba deben volverse a aplicar y continuar hasta que no se detecten más errores.

Si la metodología previa fue seguida durante el desarrollo de los casos prueba, y si estos se han ejecutado sin causar una falla, entonces la unidad está lista para ser integrada y considerada como aprobada.

### **Pruebas de componentes que no sean programas**

Un sistema computarizado se extiende más allá de las fronteras del hardware actual y de los programas que automatizan funciones. Los procesos mecánicos, humanos y una variedad de documentación son también parte del sistema.

Estos procesos y la documentación son llamados componentes -no- programa, por lo que se debe identificar que componentes del sistema no son programas, a fin de probarlos.

El propósito de esta sección es describir los componentes no programas. Así que es necesario conocer los objetivos de estos elementos para poder saber si se han cumplido:

**~ "NO SE PUEDE PROBAR ALGO QUE NO SE HA ENTENDIDO"**

Los sistemas de información además de consistir de Hardware y Software, también incluyen los siguientes tipos de documentación y procedimientos:

FORMAS.

DOCUMENTACIÓN DE POLÍTICAS Y PROCEDIMIENTOS.

MANUAL DE PROCESOS.

ESPECIFICACIONES DE TRABAJO.

GUÍA DE OPERADOR Y DEL USUARIO.

MANUAL DE REFERENCIA.

El objetivo de la prueba de los componentes que no son programas es verificar que el material y los procesos son correctos.

Ejemplos de defectos que podrían ser descubiertos durante un elemento no-programa son los siguientes:

- Páginas numeradas incorrectamente en un manual.
- Manuales con información técnica incorrecta.
- Procesos que no funcionan.
- Procesos con pasos omitidos o pasos superfluos.

Este tipo de componentes son probados en la misma manera que los programas. Y es importante la participación del usuario, ya que se pretenderá simular que se esta en un ambiente ya productivo, y que se desconoce alguna función o proceso o secuencia a seguir, y se recurre a los documentos de apoyo, a fin de demostrar que son útiles y que ya no contienen errores.

De esta manera se completan todas las funciones y actividades que como mínimas se deben ejecutar al iniciar, y aplicarse durante el desarrollo de un sistema, y se garantiza un total control de él, bajo un costo aceptable, con lo que se puede tener la confianza de iniciar la fase de construcción sin temor a sufrir retrasos, incremento de costos u omisiones importantes.

## CAPÍTULO III

### ACTIVIDADES OPCIONALES DE VERIFICACIÓN Y VALIDACIÓN

En este capítulo se revisarán algunas actividades, que aunque no son totalmente indispensables para probar un sistema y que por si mismas no garantizan que una entidad o elemento estén libres de errores, son de gran utilidad y de alto beneficio cuando se quiere lograr alta calidad en el desarrollo de cada etapa.

Solo en circunstancias muy especiales y condiciones adversas, es cuando se podrían omitir algunas de estas actividades, pero sin embargo, se debe hacer un esfuerzo para cumplir con estas otras alternativas, que en el mediano y largo plazo garantizan que fue la mejor decisión el haberlas seguido.

#### I.-) Registro, seguimiento y resolución de problemas

Los esfuerzos destinados a probar el sistema están basados en las metodologías usadas, que sin lugar a dudas deben ejecutar las siguientes funciones :

- Seguimiento a problemas y reporte de los mismos.
- Análisis de defectos.
- Control de cambios.

Los esfuerzos invertidos solo funcionarán si estos tres puntos han sido observados en cualquier fase del sistema, ya que el seguimiento y reporte determina los problemas que ya están resueltos y cuáles quedan pendientes, además de que hace fácil determinar el status actual del proyecto o de un problema en particular, mantener la historia y frecuencia de los defectos y consecuentemente documentar también las soluciones.

El reporte de problemas debe iniciar tan pronto como se inicie la primer fase del ciclo de vida del sistema y continuar durante toda su vida. Los datos pueden ser obtenidos del análisis de los programas, revisiones, pruebas unitarias formales y de todos los niveles de prueba vistos en el capítulo dos.

Se ha identificado que los problemas pueden ser resueltos de una de las tres maneras siguientes:

1. Determinando que lo que aparece como un problema, realmente no lo es. Esto podría significar que los casos de prueba que detectaron un problema estén en un error, es decir, que alguna persona que reporto el problema no entendió alguna función del sistema.
-

2. Identificar el error y corregirlo.

3. Identificar el error y posponer su solución. En este caso se tiene que incluirlo en la información del control de cambios.

De cualquier manera, se puede observar que pueden existir varias razones para probar algo: porque es nuevo, porque sufrió un cambio o porque tuvo un error. Pero algo que es evidente es el identificar lo que se quiere probar y por ello un reporte de problemas debe contener:

- **Identificación del problema.**

Consiste en un identificador único y una descripción del problema.

- **Fuente del Reporte. (quién lo envió.)**

- **Severidad del problema.**

Se debe definir una escala en donde se indique la gravedad del problema. Con esta escala se ayudará a definir la prioridad al resolver problemas.

- **Estatus del problema.**

Esta información permite seguir la huella del problema del reporte hasta su solución.

- **Acción tomada.**

Se define qué se hizo para resolver el problema, cuál fue el impacto, módulos involucrados, etc.

- **Categoría del defecto.**

Incluye omisión de requerimientos, errores de interfase, errores lógicos, etc.

- **Historia del problema.**

Aquí se proporciona un antecedente y secuencia de hechos que fueron originando el problema y cómo éste se creó.

Un reporte solo será cerrado cuando se corrigió.

A continuación se muestra un ejemplo de cómo los defectos podrían clasificarse para su mejor aplicación.

## LISTA DE CATEGORÍA DE DEFECTOS

Seleccionar una de las categorías de la lista. Escribir el código de la categoría en el cuadro del reporte de problema.

<b>CÓDIGO</b>	<b>DESCRIPCIÓN</b>
<b>Requerimientos</b>	
R1	Requerimientos incompletos o erróneos.
R2	Requerimientos incompletos.
R3	Requerimientos contradictorios.
R4	Especificaciones de interfase exterior errónea.
R5	Especificaciones del usuario no entendidas o reflejadas en los requerimientos.
R6	Factores de calidad como factible, mantenible, usable y flexible no especificados.
<b>Diseño</b>	
D1	Errores lógicos o secuenciales.
D2	Entrada de datos no validados o editados correctamente.
D3	Funciones perdidas.
D4	Errores en el almacenamiento de datos.
D5	Algoritmos inapropiados.
D6	Errores de interfase interna.
D7	Errores de redondeamiento o transacción
D8	Errores de índice.
D9	Error de ciclado.
D10	Manejo inapropiado.
D11	Error de desempeño o ejecución.
D12	Especificaciones de requerimientos mal entendidas.
<b>Construcción</b>	
C1	Decisiones lógicas incorrectas.
C2	Cálculos incorrectos.
C3	Error de ciclado.
C4	Errores en el lenguaje o sintaxis usada.
C5	Errores de índice.
C6	Error en la información.
C7	Errores en la interfase.
C8	Error de inicialización o declaración.



Soporte en la producción:

P1	Errores de documentación.
P2	Errores de capacidad.
P3	Errores de desempeño o ejecución.
P4	Errores en la recuperación de desastres.
P5	Errores en la información.
P6	Errores en el medio ambiente.
P7	Error en la secuencia de calendarización/programación.

## II.-) Análisis de defectos.

Cuando al desarrollar una nueva aplicación se ha tenido la consideración de contar con un sistema de control de cambios que refleje la razón de las modificaciones, frecuentemente se encontrará con que muchos defectos ocurren en fases comunes a pesar de que su origen fue en una fase previa o diferente.

Los datos obtenidos en el sistema de registro y reporte de problemas pueden ser analizados si estos han sido previamente tabulados.

Este análisis de defectos ayudará a los participantes del desarrollo del sistema a identificar los siguientes aspectos:

- El porcentaje de defectos que fue introducido en cada fase.
- La categoría de los defectos más comunes.
- Qué resultados, fases o productos son los más defectuosos.
- Qué defectos costaron más al ser depurados.

El efectuar y llevar a cabo este análisis, ayudará a prevenir errores en futuras aplicaciones y en las fases siguientes.

El principio de PARETO dice que el 80 % de los problemas ocurre en el restante 20 % del sistema. Es decir, los defectos se juntan debido a que son consecuencia de un error que los provocó previamente, por lo que al identificar los procesos y resultados más defectuosos es cuando se logran identificar los mayores problemas.

***“Siempre se debe recordar que los defectos cubren a otros y que éstos se juntan en alguna fase del sistema”***

### III.-) Independencia en las pruebas.

La independencia en las pruebas es un concepto que muy pocas veces es utilizado, y esto es debido a que generalmente cada desarrollo no goza del tiempo suficiente para ser aplicado. Esta independencia es subsecuente a las pruebas que se realizan del componente en turno.

Cuando una aplicación esta lista para probarse, generalmente es una persona o un grupo que participó en el desarrollo, quien efectúa las pruebas y estas están caracterizadas por un íntimo conocimiento del diseño e implantación del sistema.

Sin embargo, estas pruebas llamadas dependientes presentan el inconveniente de que se crean prejuicios respecto a la funcionalidad del sistema. Cuando una persona ha creado algo y desea verlo funcionar, diseña pruebas que van muy orientadas a demostrar que el sistema realmente funciona y muy difícilmente a encontrar errores. Esto debido a que existe un sentimiento de protección que puede generarse al actuar como un detector de errores.

Estas pruebas presentan la ventaja de que quien las aplica conoce el sistema y lo que debe hacer, pero pueden generarse omisiones por el hecho de estar inmerso en todo el proceso, al que pueden influir las presiones del usuario, del tiempo e inclusive la urgencia de concluir y liberarlo.

Una manera de reducir estas influencias negativas, es el seleccionar a un grupo para que pruebe el sistema o módulo, pero este grupo debe ser ajeno al desarrollo del sistema, y no haber participado previamente en ninguna actividad. De esta forma los prejuicios respecto a la finalidad del sistema se reducen enormemente, pero continuarán abiertas algunas posibilidades de dar por hecho algunos supuestos o hechos, que han sido adoptados por la compañía y que son de uso general.

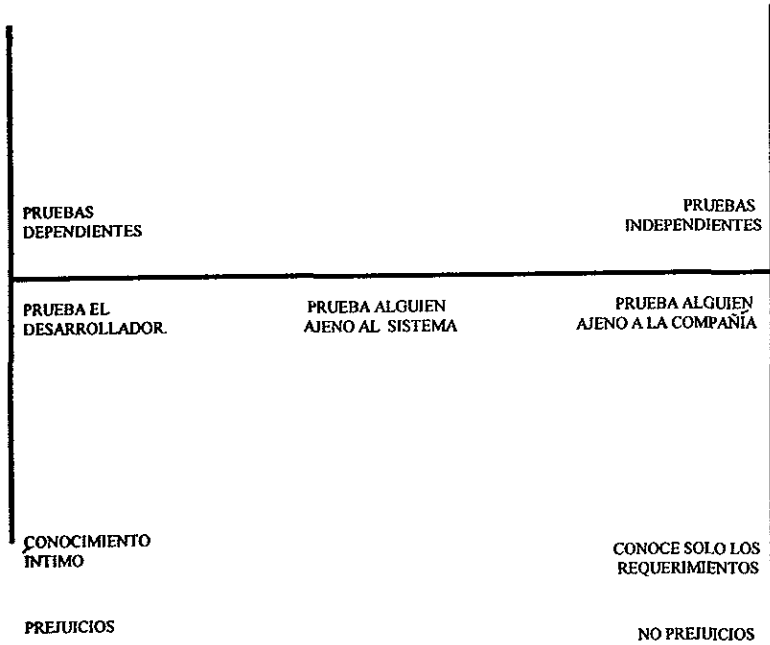
***“La propuesta es aplicar pruebas independientes sin eliminar las pruebas dependientes, lo que permitirá pasar de una alta dependencia a una alta independencia.”***

Esto se ilustra en la figura III-1 que se muestra más adelante.

Estas pruebas son efectuadas por personas ajenas a la organización y consecuentemente al sistema. Se caracterizan solo por el conocimiento de los requerimientos funcionales, y de esta manera los prejuicios no son un problema. Pero las pruebas dependientes no deben evitarse pues el agente externo no conoce el sistema como el creador. Esta falta de conocimiento puede evitar que se ejecuten casos prueba muy interesantes.

Se debe considerar que el involucramiento del grupo independiente de pruebas debe empezar lo antes posible durante las fases de definición y análisis, ya que así se podrá efectuar la planeación de las pruebas.

## INDEPENDENCIA EN LAS PRUEBAS



ES MÁS FÁCIL ENCONTRAR ERRORES CON  
GENTE EXTRAÑA AL DESARROLLO DEL SISTEMA.

FIG. III-1

Cuando se llega a la fase de diseño de negocios y prácticamente se está en el umbral del diseño técnico, los planes para las pruebas unitarias deberán ser efectuados, de tal manera que cada unidad, al menos en forma independiente, tenga casos prueba a ser aplicados para garantizar su correcto funcionamiento.

#### **IV.-) Pruebas Dinámicas.**

Una actividad optima es el aplicar pruebas dinámicas de los componentes NO-PROGRAMA, que consisten en simular un medio ambiente similar al que existirá cuando el componente esté concluido.

Por ejemplo, la guía de instalación puede ser probada dinámicamente teniendo a varias personas tratando de usarla para instalar alguna aplicación.

Debido a que es imposible y demasiado caro tratar de crear una simulación completa y real de cada ambiente, se debe tener mucho cuidado en el diseño de la simulación y para ello es de gran importancia lo siguiente:

- Elegir a participantes que serán los usuarios finales y típicos en cuanto a experiencia y entrenamiento.
- Escoger gente con experiencia en sistemas. Esta gente deberá tener mucha, regular y poca experiencia.
- Seleccionar elementos esenciales en el ambiente que se va simular.

Cuando se ha logrado obtener el ambiente, los participantes deberán usar los componentes NO - PROGRAMA, observar y registrar cada evento y anomalía a fin de detectar cualquier error u omisión.

#### **IV.A.-) Material de Entrenamiento**

El material de entrenamiento es probado directamente y dinámicamente con pruebas piloto. La efectividad del curso es medida observando si los alumnos saben hacer lo que se supone que deben aprender. La mejor, pero más difícil, manera de saber si ellos pueden ejecutar la actividad asignada, es observándolos directamente hacerlo. Para ello se requiere una sala de instrucción, un instructor y estudiantes.

#### **IV.B.-) Manuales de usuario y operación.**

Los manuales de usuario y operación son probados por un grupo de usuarios y/o operadores leyendo el manual. Mientras tanto el líder del proyecto o el ingeniero en sistemas debe observar, tomar notas y hacer preguntas que le confirmen la utilidad del manual.

Esta simulación requiere de un salón tranquilo con el equipo apropiado. Se le puede solicitar al operador o usuarios ejecutar tareas típicas, y ver como lo hacen.

#### **IV.C.-) Manual de Referencia.**

Los manuales de referencia pueden ser probados buscando un tópico en particular. Se debe tener la consideración de que los manuales de referencia no se leen de principio a fin.

Generalmente, el ingeniero en sistemas debe generar una serie de problemas y ver como los participantes los solucionan apoyándose en ese manual.

*Estas técnicas opcionales de prueba, frecuentemente son omitidas, debido a que exigen condiciones especiales para su aplicación, lo que necesariamente implica contar con mayores recursos en tiempo y dinero. Sin embargo, la decisión de aplicarlas o no, deberá tomarse después de comparar el costo que podría significar contra el costo de no aplicarlas, asociado directamente con la dimensión y alcance del proyecto en la organización de que se trate.*

## CAPÍTULO IV

### CRITERIOS PARA ESTABLECER UN CONJUNTO DE PRUEBAS PARA UN SISTEMA

#### **Introducción.**

Los esfuerzos encaminados a elaborar un conjunto de pruebas son parte de las muchas funciones que se deben realizar para crear un sistema de gran calidad. Sin embargo, cuando se presenta la oportunidad de efectuar pruebas, se debe seguir una serie de criterios, los cuales deben haber sido definidos con toda anticipación, a fin de evitar improvisar con su correcto funcionamiento.

Se debe recordar que no es de interés único probar la correcta operación de las funciones o actividades para las que el sistema fue ideado, sino que además de ello, deben incluirse algunas otras posibilidades y acciones que podrían darse al momento de utilizarlo.

Es fundamental definir los criterios con los cuales se quiere medir y probar un sistema, de tal forma que estos deben ser totalmente planeados. Para la definición de los criterios es necesario considerar lo siguiente:

- Definir procedimientos de verificación.
- Sugerir un ejemplo de los procedimientos de verificación.
- Definir una autoevaluación.
- Definir las pruebas estáticas y dinámicas.
- Reconocer la existencia de algunas técnicas para crear casos de prueba.

#### **L-) Procedimientos de verificación.**

Un ejemplo de un procedimiento de verificación, es el uso de un compilador de sintaxis, ya que puede ayudar a prevenir el inapropiado uso de un lenguaje de programación.

La definición de procedimientos de verificación es el método más conveniente porque previene defectos en los sistemas desde un principio.

## **II.-) Autoevaluación.**

La autoevaluación es todo lo que un ingeniero en sistemas puede hacer para evaluar la calidad de un producto antes de que nadie más lo vea. La autoevaluación puede ser considerada como una prueba privada, misma que incluye cuatro aspectos primordiales:

- **Lista de chequeos.**

Es una lista de los errores cometidos con mayor frecuencia, que el ingeniero de sistemas usa como una ayuda para recordarlos.

Estas listas pueden encontrarse en una variedad de libros, incluyendo las guías del sistema. Cada programador o analista puede crear una lista personal que refleje los errores más comunes cometidos por los programadores.

- **Pruebas de escritorio.**

Es una técnica para descubrir defectos en la programación. Es una lectura detallada de un producto, buscando encontrar errores de lógica y de sintaxis. Es altamente recomendado para funciones que el ser humano realiza muy bien y que la computadora no puede hacer tan eficiente. Casos que la computadora hace muy bien, como una rutina matemática complicada, deben dejarse solo para la computadora.

Si se cuenta con herramientas automatizadas, éstas pueden ser más eficientes que las pruebas de escritorio.

Para la revisión de sintaxis y de referencias cruzadas se debe leer y analizar línea por línea del producto y hacer referencia cruzada en forma individual. Es necesario invertir un poco de tiempo, pero este será recompensado encontrando errores con bastante anticipación.

Para la adecuación de las especificaciones con los estándares de calidad requeridos, se deben crear las especificaciones desde el producto mismo y comparar la complejidad del programa con la del sistema.

Una buena técnica de prueba de escritorio es el rehacer la especificación a partir de lo que el código del producto dice o muestra, y compararla con la especificación original. Si hay una buena identificación y coincidencia es una excelente señal de que la programación se hizo correctamente.

---

- **Pruebas unitarias.**

La prueba unitaria es una prueba hecha producto por producto, antes de que estos sean integrados y el sistema probado. El número de defectos encontrados y corregidos durante esta prueba no son registrados.

- **Revisión informal por parejas.**

Esta revisión se da cuando un ingeniero de sistemas recibe ayuda de un compañero en el proceso de descubrir y corregir errores. Al igual que la prueba unitaria, esto es considerado privado y no es oficialmente registrado. Este tipo de revisión es altamente eficiente para la detección de errores, ya que es muy similar a cuando alguien que tiene un problema técnico, lo comenta con alguien más y se encuentra la solución.

### **III.-) Pruebas estáticas.**

Las pruebas estáticas son aquellas donde no se ejecuta absolutamente nada; de ahí su nombre. Es exclusivamente un proceso humano, en donde se comparan el producto con sus requerimientos y especificaciones.

La intención es verificar la coherencia existente entre un producto ya creado y las definiciones originales que tuvo, a fin de que sin incluir recursos mayores, como: procesamiento, consumo de papel y otros insumos, se puedan detectar discrepancias en su programación. Se efectúa la comparación para determinar si se tienen que trasladar las operaciones del sistema de una fase a otra.

El resultado es comparado así mismo y se evalúa el proceso usado para identificar si los estándares internos y la consistencia se han mantenido.



Es por ello que las pruebas estáticas constan de:

### **III.A.-) Revisiones.**

La revisión es una examinación de grupo de un producto del sistema. En una revisión los objetivos principales del producto son discutidos por el grupo. El puro acto de discutir ayuda a encontrar algunos problemas, y esto está basado en que los revisadores hacen preguntas, aclaran aspectos y detectan áreas con posibles fallas.

Algunos de estos problemas que generalmente son identificados son :

- Contradicciones entre el producto y sus especificaciones.
- Ambigüedad en el diseño del uso de la interfase.
- Omisión de funciones requeridas.
- Ineficiencias en el código o en el diseño.
- Asumir o presuponer erróneamente.
- Estar fuera de estándares.

### **III.B.-) Inspecciones.**

Durante una inspección el desarrollador narra la lógica de la línea que inspecciona. Se generan preguntas y los problemas son expuestos. Y aquí es cuando las listas de errores más frecuentes tienen gran utilidad.

### **III.C.-) Auditorías.**

Una auditoría es la verificación de que las tareas requeridas para una actividad han sido concluidas. Esto confirma que la documentación está completa, que las revisiones requeridas, inspecciones y las pruebas han sido ejecutadas y que todos los problemas han sido reportados y resueltos.

Uno de los beneficios de efectuar auditorías es que los desarrolladores saben que existirán, y si ellos saben que lo que están generando será revisado, tratarán de asegurar que se encuentre completo. Estas auditorías no se concentran en problemas técnicos o explicaciones, sino en lo que existe y está completo, y deben aplicarse al final de cada fase del ciclo de vida de los sistemas.

#### **IV.-) Pruebas dinámicas.**

Las pruebas dinámicas confirman que el desarrollo de la aplicación se encuentre acorde con las especificaciones. En contraste con las pruebas estáticas, están basadas en la ejecución de los programas o por la simulación de un proceso no automatizado. Nuevamente es comparada la funcionalidad de la aplicación con sus requerimientos y la estructura, con los estándares internos.

Una planeación oportuna de pruebas dinámicas proporciona como beneficios el asegurar la disponibilidad de los recursos en el momento en que estos son necesarios y la corrección a tiempo de errores en la programación.

#### **V.-) Criterios para establecer un conjunto de pruebas.**

El proceso de prueba es una parte integral de las actividades de control de calidad que deben efectuarse durante el desarrollo de un sistema.

A continuación se definen los criterios que ayudarán a definir un conjunto de pruebas e identificar puntos de control, productos y atributos a ser controlados.

#### **PASOS PARA DETERMINAR LOS CONJUNTOS DE PRUEBA**

1. Identificar los procesos a controlar.
2. Identificar los puntos de control.
3. Identificar los productos o resultados.
4. Identificar los atributos
5. Identificar los estándares y criterios
6. Ejecutar las actividades que crean las salidas.
7. Medir y Comparar.
8. Analizar la tendencia de los defectos.

# PROCESO DEL DESARROLLO DE LA ESTRATEGIA DE PRUEBAS

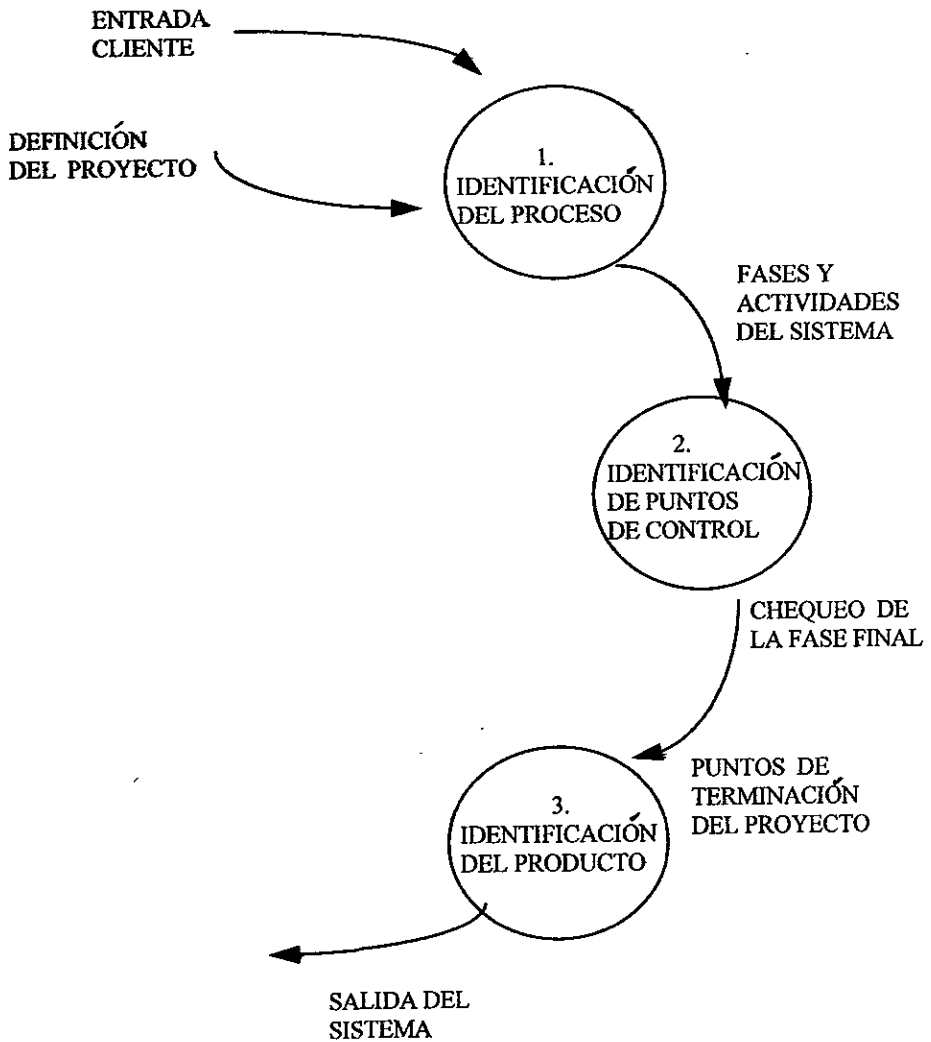


FIG. IV-1

En el diagrama de flujo anterior se ilustra el proceso de desarrollo de criterios de prueba. Los círculos indican los procesos y las flechas indican el flujo de datos; las salidas de un proceso son entradas del siguiente. De tal forma que el proceso de desarrollo de la estrategia de pruebas se compone de ocho pasos que se ilustran en las figuras IV-2 y IV-3, y que se describen a continuación:

### **1. Identificar los procesos a controlar.**

El proceso de definición de criterios de prueba se inicia definiendo cuáles se van a controlar. Esto se hará tomando en cuenta las necesidades del cliente, la definición y la historia de las fases del sistema.

### **2. Identificar los puntos de control y actividades a ser realizadas.**

Estos puntos incluyen los puntos en el calendario en donde los resultados del sistema se deben completar, tales como especificaciones de requerimientos, de diseño y código.

### **3. Identificar los productos que estarán disponibles para pruebas.**

Es importante que todos los productos que se probarán cuenten con los procesos necesarios para efectuar la prueba lo más completa posible, ya que si un módulo o programa requiere de una salida previa, ésta esté disponible para aprovecharla, de lo contrario se causará que no se efectúe completa o correctamente.

### **4. Identificar los atributos de medición de esos productos**

En este punto se determinan los atributos de los productos que se desean medir, estos incluyen aspectos como son: el verificar que la función este completa, cuál es la estructura de la corrección, eficiencia, mantenimiento y atributos de desarrollo.

Los atributos están comúnmente referidos a las HABILIDADES que el sistema debe tener, porque incluyen cosas como reusabilidad, probabilidad y mantenimiento.

### **5. Identificar estándares y criterios contra los que se miden los atributos.**

Los estándares por los que los atributos serán evaluados son proporcionados por los requerimientos de cada producto, la cobertura de criterios, complejidad, estándares internos y procedimientos.



## PROCESO DEL DESARROLLO DE LA ESTRATEGIA DE PRUEBAS.

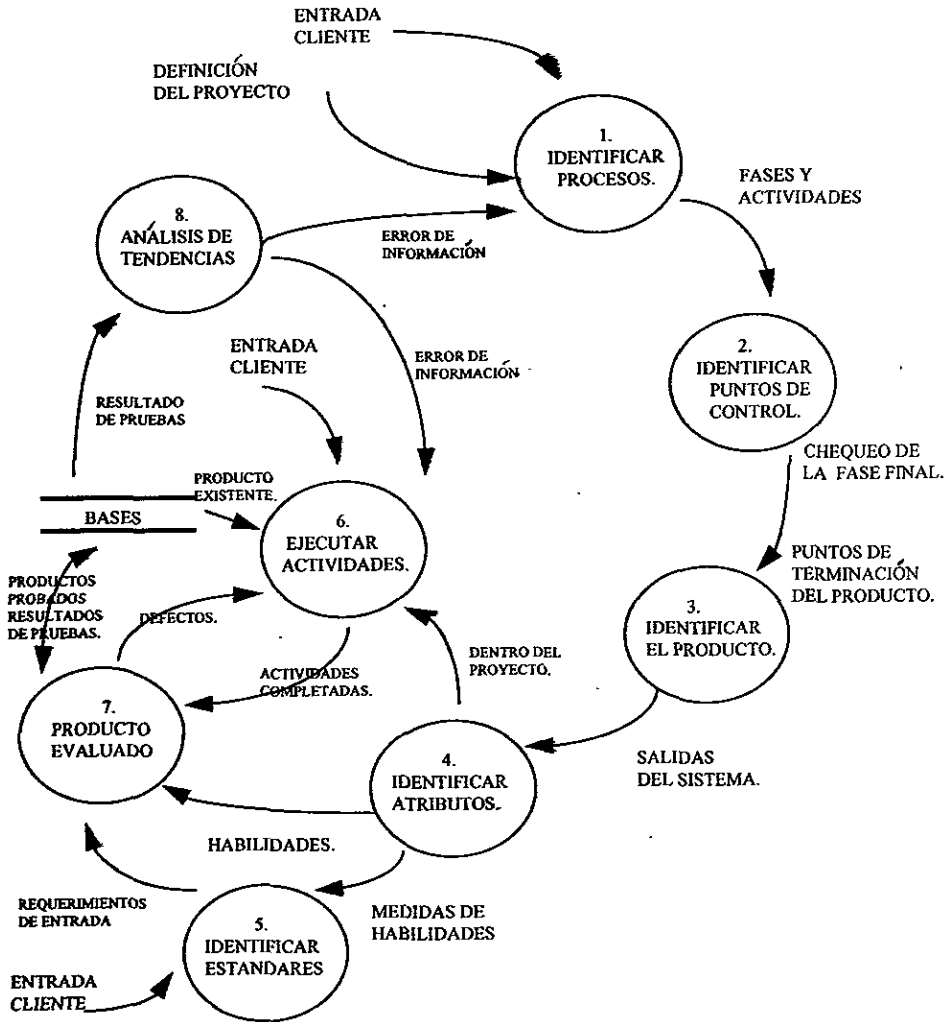


FIG. IV-3

## **6. Ejecución de las actividades del sistema en cada fase para crear salidas o productos.**

El sistema es ejecutado de acuerdo con los planes establecidos por el proyecto y los requerimientos del cliente. Esto puede ser un proceso interactivo donde las actividades son re-ejecutadas para corregir defectos.

## **7. Medición de atributos seleccionados y compararlos contra los estándares establecidos.**

Los atributos de cada resultado o producto son medidos y comparados nuevamente con los estándares. De esta manera el proceso de prueba será de alta calidad, puesto que es aquí donde los defectos deben detenerse y no permitir que se integren a las funciones globales del sistema, lo que sin duda imprime un grado muy aceptable de confiabilidad al sistema.

Los problemas son reportados y las funciones y operaciones del sistema son re-ejecutados para resolverlos. Los productos probados, la documentación de pruebas y los resultados son mantenidos y controlados como parte de las bases del sistema.

Actividades de medición:

- pruebas estáticas:
  - revisiones
  - inspecciones
  - auditorias
  
- pruebas dinámicas:
  - unidades de prueba
  - prueba de sistema

## **8. Analizar la tendencia de los errores y ajustes.**

Los resultados de las pruebas son analizados en términos de la efectividad de las pruebas y los procesos desarrollados.

Esta información de errores es usada para ajustar las metodologías que estén siendo ejecutadas, el proyecto y los planes de prueba. Esta información debe estar en tiempo para usarse en el momento necesario, como se muestra en la figura VI-3

## **VI.-) Puntos de control y atributos que son controlados.**

Una vez que se ha definido que unidades o procesos se deben probar, lo importante a estas alturas es identificar los puntos de control en los que se deberán analizar los resultados, cualidades y atributos con los que se debe cumplir.

Los puntos de control pueden ocurrir al momento de:

- La conclusión de una actividad importante.
- Que una decisión se deba tomar
- La terminación de una fase.
- La transferencia de información de una persona a otra, o de un grupo.
- Al momento de que la firma del cliente o usuario es necesaria.

Sin embargo, deben estar disponibles los productos a los cuales se deben aplicar los puntos de control. El hecho de crear los planes de prueba lo antes posible al desarrollo del sistema ayuda a asegurar que los recursos probados estén programados y disponibles cuando son requeridos.

El planear las pruebas, ayuda a prevenir muchos errores que se van produciendo durante el proceso de desarrollo. Mientras el desarrollo de casos prueba se va efectuando, el personal del equipo de prueba revisa la estructura y funcionalidad de los módulos y de esta manera problemas potenciales son detectados.

La creación del sistema no debe estar basado en los casos de prueba. En otras palabras, el sistema no debe desarrollarse solo para aprobar las pruebas planeadas, ya que si bien esto garantiza el funcionamiento del sistema, este será solo bajo ciertas circunstancias perfectamente definidas y absolutamente rígidas.

Los casos de prueba se han de desarrollar independientemente del sistema y pueden basarse en los mismos requerimientos del sistema. Una estrategia general para el desarrollo de casos prueba, es iniciar con los requerimientos basados en algunas técnicas, tales como la graficación de causa-efecto, particionar equivalencias, análisis de fronteras y la investigación de errores.

Todos los casos de prueba basados en los requerimientos deberán ser desarrollados y tal vez ejecutados antes de que cualquier caso de prueba basado en lógica sea desarrollado.

***"NO SE PUEDE PROBAR LA CALIDAD DE UN PRODUCTO, SÓLO SE  
DESCUBRE CUANDO EXISTE UNA POBRE CALIDAD EN ÉL."***



El resultado de probar puede ser utilizado para identificar los procesos que crearán o generarán productos con poca calidad. Estos procesos pueden ser cambiados para que produzcan procesos de alta calidad.

La siguiente gráfica se puede apreciar que las actividades de prueba se desarrollan concurrentemente con las actividades de otras fases, a fin de garantizar los mejores resultados a lo largo del Desarrollo de Sistemas.

## PRUEBAS EN EL CICLO DE VIDA DE LOS SISTEMAS

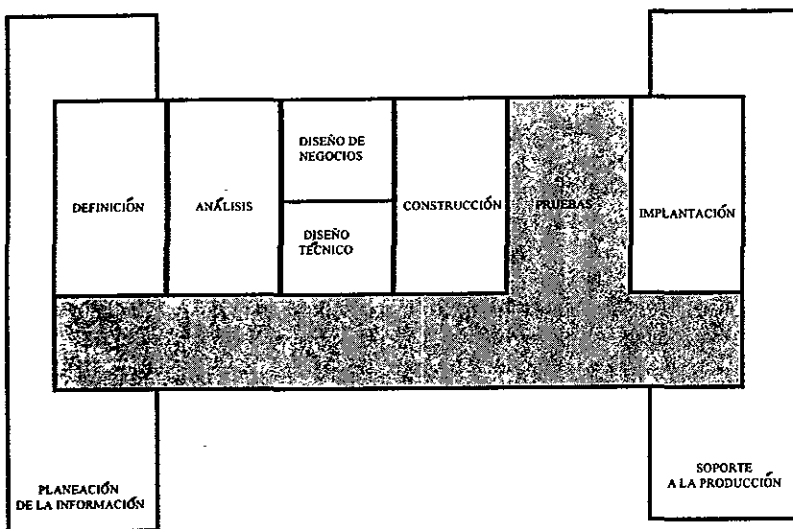


FIG. IV-4

## CAPÍTULO V

### HERRAMIENTAS Y TÉCNICAS DE PRUEBA

El número de casos de prueba para un programa o sistema puede ser infinito , y no es posible ni deseable ejecutar cada caso de prueba potencial. Las metodologías, herramientas y técnicas de prueba ayudan al Ingeniero en Sistemas en la creación de pruebas que tengan la mayor oportunidad de descubrir defectos.

En este capítulo se incluyen técnicas de prueba que permitirán diseñar casos de prueba usando los requerimientos establecidos y los basados en la lógica. El desarrollo de estos y las técnicas aplicadas caen en la categoría de pruebas dinámicas, pues debe recordarse que es en estas en donde el código es ejecutado realmente.

#### TÉCNICAS DE TIROTEO "SHOTGUN"

La peor técnica para probar es la prueba aleatoria de datos o mejor conocida como técnica del tiroteo.

Este enfoque consiste en que aleatoriamente un Ingeniero en Sistemas introduce datos al azar al sistema, lo que representa un enorme desperdicio de recursos.

***"SI SE APUNTA A NADA SE LE DARA A NADA."***

Es por esta razón que el incrementar la cantidad de pruebas al azar no compensa el diseño de una prueba eficiente, pues esto realmente incrementa la posibilidad de descubrir errores, como se vio en capítulos anteriores.

De esta manera los enfoques que deben observarse en las técnicas a probar son:

- a) Desarrollar pruebas que utilicen las especificaciones y los requerimientos sin tocar las estructuras del programa y su código.
- b) Pruebas que se dirigen a la estructura del código y que se interesan en asegurar la cobertura de múltiples condiciones.

Un beneficio inmediato de un buen caso de prueba es la reducción en más de uno de los necesarios para probar el sistema, pues encontrará defectos no descubiertos con anticipación. Descubrir el mismo defecto una y otra vez es redundante y obviamente no es un buen uso de los recursos.

## TÉCNICAS DE PRUEBA BASADAS EN LOS REQUERIMIENTOS.

Las técnicas basadas en los requerimientos son algunas veces denominadas cajas negras, debido a que se ignora la estructura o la lógica del código que se está probando. El código es visto como una caja a la cual se le introducen datos, hace algo con ellos y genera algunas salidas y resultados.

### **Técnica de Causa y Efecto:**

Las gráficas de causa y efecto son una técnica basada en los requerimientos, misma que es ejecutada en una sección del sistema para crear los casos de prueba. La técnica analiza las reacciones entre las entradas (causas) y las salidas (efectos).

Debido a que estas relaciones se complican rápidamente, se recomienda mantener la sección que se está analizando lo más pequeña posible a fin de no ampliarla y complicarla.

La elaboración de estas gráficas implica una lectura detallada y un análisis de las especificaciones, lo que tiende a exponer los problemas.

A continuación se analiza la secuencia que debe seguirse para la elaboración de gráficas de causa y efecto:

#### 1. Leer las especificaciones.

Leer cuidadosamente las especificaciones y cualquier otro documento relacionado con la sección que se va a graficar.

#### 2. Identificar las causas (entradas).

Una causa es cualquier cosa que sucede a los segmentos del sistema que se está analizando, el cual dará una respuesta. Las causas pueden ser un dato de entrada o algún evento externo.

Hay varias maneras de identificar las causas; se pueden ir subrayando cuando se están leyendo, eventualmente se pueden escribir en una lista adicional sin importar el orden.

Finalmente se obtiene la lista total de causas y las condiciones mencionadas en las especificaciones y se asigna un identificador único a cada causa.

#### 3. Identificar los Efectos (salidas).

Los efectos son las respuestas a las causas y estas pueden ser datos de salida o algún evento externo tangible. Aquí también debe obtenerse una lista con todas las condiciones o acciones de salida.











Al hacerse este análisis es de gran utilidad el cuestionarse frecuentemente ¿qué reacciones ocurren cuando el sistema funciona?.

---

4. Identificar las relaciones entre Causa y Efecto.

Aquí se relacionan cada una de las causas contra cada uno de los efectos, dibujando entre las dos listas unos símbolos que describan la dependencia entre sí. El resultado debe ser un diagrama de Causa y Efecto.

Ejemplo:

SÍMBOLO	SIGNIFICADO	EJEMPLO	DESCRIPCIÓN
	Identidad	A  E1	El si A ocurre
	No ocurre	A  E1	El si A no ocurre
	Y	A  E1 B 	El si A ocurre y B no ocurre
	No	A  E1 B 	El si A ocurre o B ocurre

5. Identificar la acción de las causas.

Observar las relaciones entre las causas para ver si alguna de ellas son mutuamente excluyentes o dependientes de otras.

Las relaciones deben expresarse en los siguientes términos:

**PALABRA CLAVE**

- Al menos uno
- Cuando más una
- Una y solo una
- Requiere
- Cubre

**EJEMPLO**

- Al menos una causa A, B, C puede existir.
- Cuando más una causa A, B existe.
- Una y solo una A, B existe.
- A requiere de B
- A cubre a B

6. Trasladar la gráfica de causa y efecto a una tabla de decisión.

La creación de tablas de decisión es más un arte que una ciencia. El objetivo es tener todas las diferentes combinaciones de las causas en los menos posibles casos de prueba. Hay más de un solo camino para lograr esto de una gráfica de causa y efecto.

Listar las causas y efectos en orden de identificador dado en el diagrama previo y colocarlos a un lado de la tabla . A manera de encabezado se ponen los identificadores de los casos de prueba.

Se usa un uno (1) para identificar que la causa o efecto está presente, y un cero (0) para indicar que la causa o efecto no están presentes. Un blanco o espacio indicará que la presencia o ausencia de la causa o efecto es irrelevante

	Caso de prueba A	Caso de prueba B	Caso de prueba C	Caso de prueba X
Causa	1	0	1	1
	1	1	0	1
	1	1	1	0
Efecto	1	0	0	1
	1	0	0	0
	0	1	1	1

Las tablas de decisión son creadas recorriendo el diagrama anterior de causas y efectos y regresando de los efectos a las causas, identificando todas las combinaciones de las causas que producen cada combinación de efectos y registrar estas combinaciones en la tabla de decisión.

Realmente la fuente de las tablas es el diagrama elaborado en el inciso 4, pues aquí se describen todos los posibles caminos que una causa tiene para llegar a un efecto. Para ello la manera en que debe seguirse el diagrama es la siguiente:

A) Se elige un efecto y se trabaja hacia atrás dirigiéndose a cada causa que produce este efecto.

B) Crear un caso de prueba para cada una de estas causas considerando:

- Si un nodo tiene el valor de 1, se necesita un caso prueba para cada una de las causas con ninguna de las otras causas presentes, de tal forma que las haya con 1 y con 0.
- Si un nodo tiene el valor de 0, entonces todas las entradas dentro de un nodo deben tener el valor 0.

C) Elegir el siguiente efecto y determinar las posibles causas que producen el efecto .

D) Continuar haciendo para cada efecto hasta que las combinaciones de causas que producen los efectos estén en algún caso de prueba.

E) Ahora se requiere examinar cada columna de caso de prueba e identificar el impacto de las causas en la columna de casos los efectos en la que la columna continúa en blanco.

La tabla de decisión se da por concluida cuando cada efecto tiene un valor de cero o uno en cada columna. Las causas pueden estar en blanco.

Otra manera un poco más rápida es generar con números binarios todas las posibilidades para el conjunto de causas, de manera tal que se combinen los Y/O entre sí, Y posteriormente se trasladan hacia los efectos.

Se analizan los efectos y se reduce a una tabla en la que se indican los efectos de combinación única en toda la primera tabla y se transcriben sus causas.

Luego se van seleccionando de las restantes combinaciones aquellas causas diferentes y se aplican para los efectos, y así subsecuentemente de manera que se obtenga una tabla menor.

#### 7. Revisión de la tabla de decisión.

Se debe asegurar que grupos idénticos de causas producen idénticos efectos o resultados. En otras palabras, si hay dos grupos de causas pero cada una muestra como resultado diferentes efectos, entonces en alguna relación hubo un error.

Sin embargo, grupos diferentes de causas podrían tener los mismos resultados.

#### 8. Generación de casos de prueba.

Cada columna en la tabla de decisión define un caso de prueba. Se debe documentar la entrada y lo que se debe esperar de cada caso, de tal forma que se deben ir haciendo las causas combinadas y checando que efectivamente se den los efectos deseados.

Después de que todos los casos de prueba están recolectados, algunos de ellos se pueden combinar y algunos otros pueden eliminarse, si otros casos ya han sido creados con entradas duplicadas.

Durante estos 8 pasos se utilizan los requerimientos para ayudar a generar los casos de prueba, se crea una gran variedad de situaciones y se ve si el sistema actua como los requerimientos lo predecian, si es que estas pruebas se efectuarán, y sino funcionan como se esperaba, es entonces que se ha detectado un error.

### **Técnica de partición de equivalencias y análisis de valores de limite.**

Debido a que muchos defectos o errores, se ocultan alrededor de los límites de cada función del sistema o interfase se requiere de un método para descubrir esos límites dentro del sistema y después analizarlos para identificar los defectos y errores hasta entonces ocultos.

Esta técnica ayudará a lograr este objetivo, pues está basada en la determinación de las diferentes entradas que el sistema deberá manejar. Una vez que se han determinado las diferentes clases de entradas, es prácticamente inmediata la detección de las fronteras o limites, donde frecuentemente existen problemas.

La segunda fase de esta técnica es la selección de valores que giran alrededor de las fronteras. Estos valores pueden ser usados en casos de prueba para probar al código y conocer como se manejan estas situaciones en los límites.

Esta técnica debe aplicarse completa; y consta de los siguientes pasos :

#### **1. Examinar las entradas.**

Las entradas para cualquier dato o campo a ser probado, pueden venir de cualquier fuente. Por ejemplo, las entradas pueden consistir de datos ingresados en una pantalla por un usuario, o quizás de la lectura a una base de datos o archivo, o de campos pasados como argumentos de una subrutina a otra.

Así, dependiendo de donde provengan los datos, será la serie de rastreos que deberán hacerse a cada una de las entradas, para identificar exactamente como se originan y como deben ser tratados.

#### **2. Determinar las clase de cada entrada.**

El principio es agrupar campos de entrada dentro de clases de equivalencia. Una clase es un conjunto de datos. Cada dato en esta clase debe ser tratado de igual manera dentro del sistema. Si los datos dentro de una clase no serán tratados igual, entonces se debe subdividir a estos campos en más clases.

Algunas veces las clases consisten de un sólo valor. Clases de datos inválidos, así como de datos válidos, deben ser identificados.

Las siguientes son algunas guías par identificar clases de equivalencia:

Si la entrada especificada esta en un rango, entonces hay que identificar tres clases de equivalencia:

- a) Una clase válida dentro del rango.
- b) Una clase inválida por abajo del rango.
- c) Una clase inválida por arriba del rango.

Si la especificación de la entrada enlista varios campos, cada uno con requerimientos únicos de procesamiento, entonces hay que identificar 2 o más clases de equivalencias:

- a) Una clase equivalente para cada tipo.
- b) Una clase inválida para cualquier caso que no sea ninguno de los esperados. Por ejemplo, un campo que puede recibir un SI o un No, deberá tener las siguientes clases:

b.1) entrada válida = SI.

b.2) entrada válida = NO.

b.3) entrada inválida = TAL VEZ UN 1 O UNA ZX.

Si la entrada es especificada como cualquier elemento de grupo de campos, cada campo del grupo debe ser tratado de igual manera, y después identificar las dos clases de equivalencias, mismas que son:

- Una clase válida para cualquiera de esos campos.
- Una clase inválida para cualquiera que no este en el grupo.

Si la especificación indica que la entrada del campo consiste en algo, entonces se identifican clases de equivalencias:

- Una clase válida que consiste en algo que cumpla la condición.
- Una clase inválida que consiste en algo que no cumple la condición.



### **3. Selección de valores de los límites de cada clase.**

Este punto está basado en la premisa de que los defectos o fallas ocurren en los límites de cada clase de equivalencia. Más que elegir un valor de en medio del rango, se debe elegir de los valores finales. Este concepto aplica no solo para datos de entrada, sino también a ciclos de ejecución, rutinas y en el proceso de tablas y archivos.

Por ejemplo, en un ciclo que se ejecuta seis veces, los errores muy probablemente ocurrirán en la primera, segunda o en la sexta vez que el ciclo se ejecute, y lo mismo pasará si el ciclo se procesa 7 u 8 veces.

### **4. Definición de casos de prueba.**

Después de determinar las clases de equivalencia y la selección de valores para los límites, se deben definir los casos de prueba a aplicarse, iniciando con las condiciones válidas, e iniciar construyendo casos de prueba válidos, usando muchos de las clases y valores de los límites ya encontrados como posibles casos de prueba. Y así consecutivamente, se deben desarrollar todos los casos de prueba válidos hasta cubrir las condiciones.

Una vez que todos los casos de prueba válidos han sido definidos, se procede a construir los casos de prueba inválidos. Cada caso de prueba inválido sólo deberá contener una condición inválida. Esto es debido a que estas tienden a encubrir otros errores; la presencia de una condición inválida puede prevenir el proceso de encontrar la segunda.

Por ejemplo, la primera condición inválida podría causar que el programa que está siendo probado, termine con un mensaje de error. En esta situación, la segunda condición inválida nunca es ejecutada, y por lo tanto, nunca probada.

Todos los casos de prueba desarrollados con esta técnica, deben ser documentados en la biblioteca de pruebas, pues algunos de estos casos podrían ser necesarios posteriormente, y así se evita la creación de nuevos o duplicados.

***“UN BUEN CASO DE PRUEBA REDUCE EN MÁS DE UNO EL NÚMERO DE CASOS DE PRUEBA NECESARIOS PARA PROBAR TODO UN SISTEMA”***

### **Técnica de adivinar los errores.**

Supóngase que sistemáticamente se han aplicado todas las técnicas de casos de prueba en forma correcta, pero que existe un sentimiento de que hay más de dos situaciones que aún deberían probarse y que no se sabe exactamente cuales son.

Estas corazonadas ocasionales o la experiencia de saber que siempre se comete un mismo error, se clasifican dentro de las pruebas adivinadas para encontrar errores.

La técnica de adivinar errores es la que nos lleva a usar la experiencia obtenida con el proyecto u otros sistemas para desarrollar casos de prueba y encontrar errores. Sin embargo, debe tenerse mucho cuidado en la documentación correcta de cada uno, para evitar redundancias y no caer en pruebas aleatorias sin sentido.

Los casos de prueba generados por esta técnica de adivinación incluyen lo siguiente:

- Problemas que la persona que esta probando haya tenido en el pasado, pues su experiencia le ayuda a descubrir defectos.
- Errores que la persona que hace la prueba ha cometido, pues conoce sus puntos débiles y puede compensarlos.

Y como estos errores pueden haber muchos, sin embargo, no se debe caer en un trabajo a base de prueba y error, sino que se debe estar sustentado principalmente en experiencias reales y previas.

## **PRUEBAS BASADAS EN LA LÓGICA.**

Las técnicas basadas en la lógica son frecuentemente denominadas técnicas de cajas blancas, debido a que se busca en el código que se desarrolla para definir casos de prueba. Se trata de probar y ejercer el código a fin de lograr la cobertura deseada.

### **1) Gráficas para el control del flujo.**

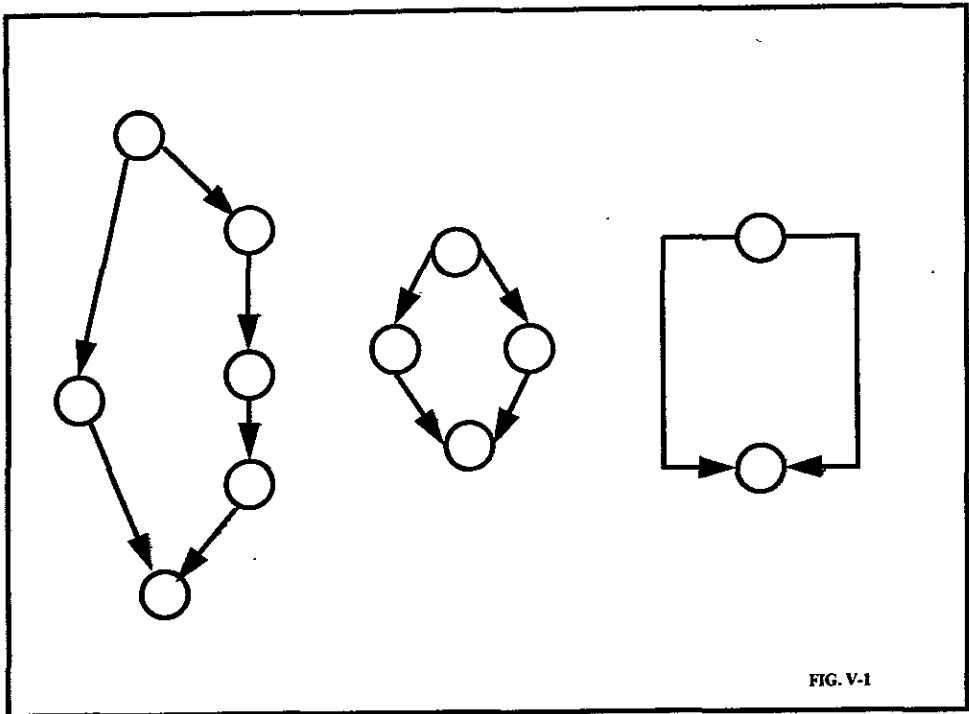
Estas gráficas son representaciones del flujo del código, estas hacen la estructura de un programa más obvia y útil para el analista que desarrolla los casos de prueba.

La elaboración de las gráficas sigue algunas reglas:

- Deben tener un único nodo inicial.
- Deben tener un único nodo final.
- Cada nodo en la gráfica debe ser localizable desde el nodo inicial.
- Cada nodo en la gráfica debe ser capaz de llegar al nodo final.

Los nodos que tienen sólo una salida (flecha) son nodos secuenciales que pueden ser eliminados y no cambian el significado de la gráfica.

Ejemplo:



Esta gráfica puede ser simplificada eliminando los nodos que fluyen secuencialmente.

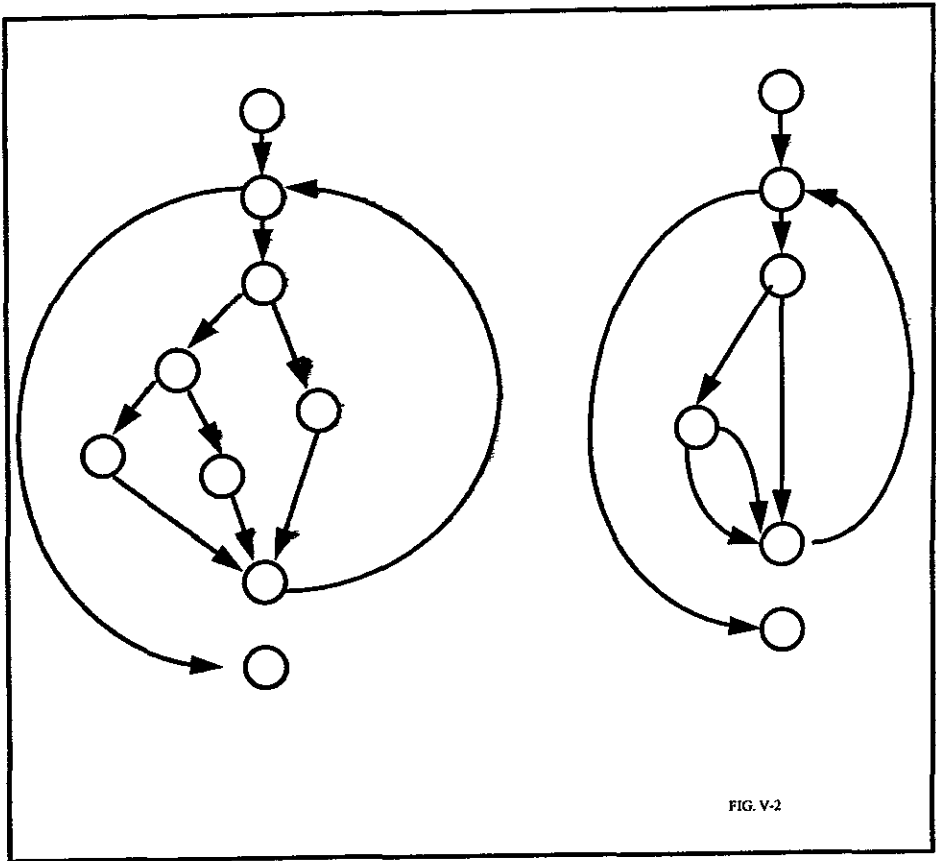


FIG. V-2

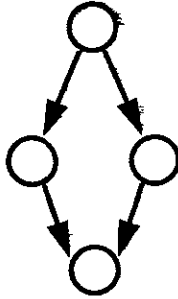
A continuación se muestran las gráficas de las construcciones de programación más comunes,

# PROGRAMACIÓN COMÚN CONSTRUCCIONES.

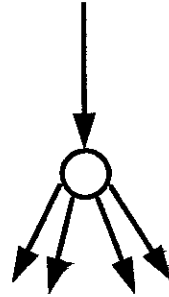
Secuencia



Si- Entonces- O



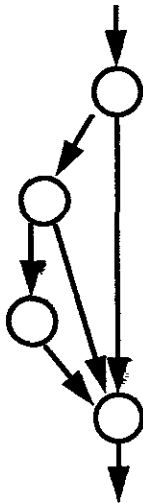
En - Caso - De



Si - Entonces



Si se compone de Y



Si se componen de O

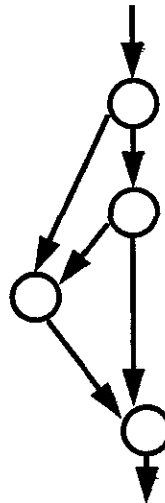
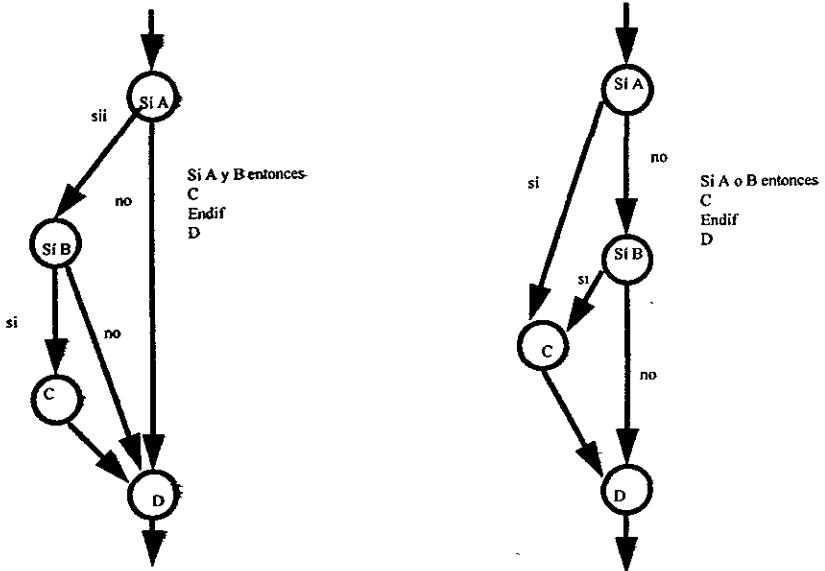


FIG V-3



Los círculos de decisión pueden ser representados en diferentes formas:

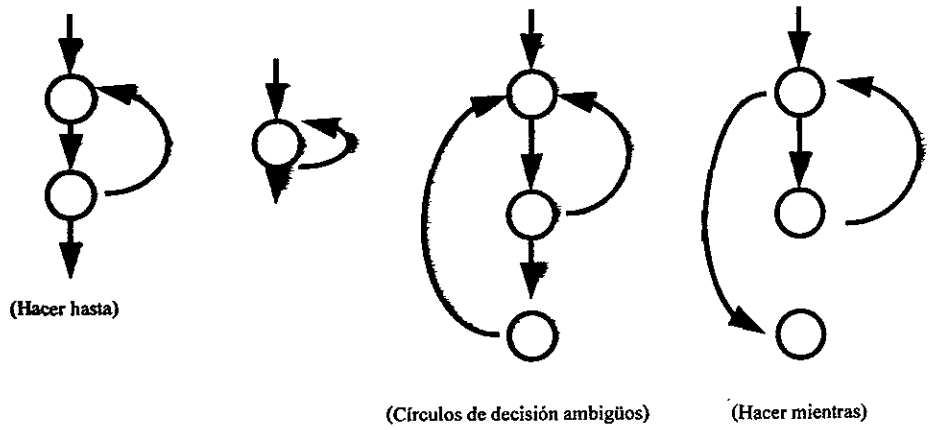


FIG. V-4

Como puede verse después de obtener la gráfica de flujo del código, se deben etiquetar cada uno de los nodos y las flechas o conectores para que puedan ser referenciados fácilmente.

Los nodos son etiquetados con números.

Las flechas son etiquetadas con letras minúsculas.

Sólo los nodos que son decisiones son etiquetados con letras mayúsculas.

Si la condición es verdadera, la flecha se etiqueta con la letra mayúscula que tiene el nodo que es la decisión. Si la condición es falsa, la flecha se etiqueta con una raya arriba con la letra mayúscula del mismo nodo de la decisión.

Son los nodos y las flechas que afectan al flujo las que se etiquetan, y los nodos que no son esenciales no se etiquetan y estos son aquellos que son secuenciales.

Con estos diagramas se tiene mayor información para determinar que se probará y hasta que punto se cubrirá, de tal forma que se tenga un código confiable.

## **2) Análisis de Cobertura.**

Las gráficas de flujo se utilizan para generar casos de prueba que permitan hacer una suficiente cobertura, pues se debe recordar que es imposible poder probar todo. El criterio de cobertura es especificado en el plan de pruebas.

El criterio de cobertura que podría ser especificado en el plan de pruebas es el siguiente:

- Cada declaración en la unidad de prueba es ejecutado al menos una vez.
- Cobertura de Decisiones.  
Cada rama en el programa es utilizada al menos una vez. Esta también implica que cada declaración es ejecutada al menos una vez.
- Cuando múltiples nodos de entrada y condiciones son utilizadas, la persona que prueba debe asegurar que esto sucede, ya que debe obligar a que pase por todas ramas.

Cuando se especifica en el plan de pruebas una cobertura de decisión se asume que la cobertura de declaraciones o condiciones también será incluida.

Los casos generados por las técnicas basadas en la lógica complementan a los casos de prueba creados anteriormente, utilizando técnicas basadas en los requerimientos.

Antes de generar casos de prueba basados en lógica, se debe determinar que caminos se han cubierto anteriormente por otros casos de prueba.

**Ejemplo:**

Se tiene una unidad que se desea probar y consta del código siguiente:

```
IF (AAA > 1) AND (BBB = 0) THEN  
XXX=CCC/AAA  
ENDIF  
IF (AAA=2) OR (CCC > 1) THEN  
XXX=CCC + 1  
ENDIF
```

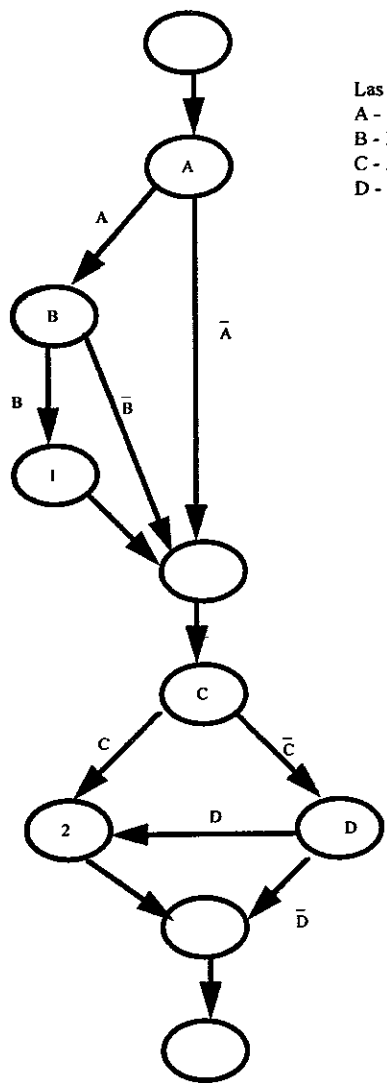
Y se requiere cobertura de decisiones y declaraciones. Desarrollar casos de prueba para esta unidad.

**Solución**

El diagrama de flujo para esta unidad se debe dibujar como se muestra en la siguiente lámina:



**Ejemplo de Condición Múltiple.**



Las condiciones son:  
 A - AAA > 1  
 B - BBB = 0  
 C - AAA = 2  
 D - CCC > 1

Si (AAA > 1) y (BBB = 0) entonces  
 XXX = CCC/AAA  
 Endif  
 Si (AAA = 2) ó (CCC > 1) entonces  
 XXX = CCC + 1  
 Endif

FIG. V-5

Ahora bien, partiendo de esa base se deben elaborar los casos de prueba. Para ello es necesario crear una tabla de decisión para crearlos de manera que permitan probar todas las combinaciones que hagan una decisión compuesta.

a) IF (AAA > 1) AND (BBB = 0)

AAA > 1	0	1	1	0
BBB = 0	1	0	1	0

misma que se extrae de la tabla binaria:

AAA > 1	BBB = 0
0	0
0	1
1	0
1	1

b) IF (AAA = 2) OR (CCC > 1)

AAA = 2	0	1	1	0
CCC > 1	1	0	1	0

misma que se extrae de la tabla binaria:

AAA = 2	CCC > 1
0	0
0	1
1	0
1	1

Puede parecer trivial probar la combinación  $\bar{A}B$  y  $A\bar{B}$  en la tabla 1, o sea: ( $AAA \leq 1, BBB = 0$ ) y ( $AAA \leq 1, BBB \text{ NOT} = 0$ ), ya que no hay camino que pueda generar que se ejecute esta combinación.

Igualmente esto podría parecer para la combinación  $CD$  y  $\bar{C}\bar{D}$  de la segunda tabla. Sin embargo, probar estas combinaciones expondrá cualquier error de la lógica en la expresión compuesta. Las pruebas se hacen para asegurar que el árbol de decisión está estructurado correctamente.

Es altamente recomendable utilizar múltiples condiciones:

$AB$	$AAA > 1$	$BBB = 0$
$\bar{A}B$	$AAA \leq 1$	$BBB = 0$
$A\bar{B}$	$AAA > 1$	$BBB \text{ NOT} = 0$
$\bar{A}\bar{B}$	$AAA \leq 1$	$BBB \text{ NOT} = 0$
$CD$	$AAA = 2$	$CCC > 1$
$\bar{C}\bar{D}$	$AAA \text{ NOT} = 2$	$CCC > 1$
$C\bar{D}$	$AAA = 2$	$CCC \leq 1$
$\bar{C}D$	$AAA \text{ NOT} = 2$	$CCC \leq 1$

Después de identificar las condiciones se identifica la relación entre estas.

Existen dos tipos de relaciones

#### a) Dependencia de Proceso.

- Una variable de entrada es un "proceso dependiente", cuando el valor de la variable es modificado por el proceso.
- Una condición es un "proceso dependiente" cuando su valor verdadero es modificado por el proceso.
- En viceversa, aquellos que no se modifican son "procesos independientes".

#### b) Correlación.

Dos o más variables de entrada o condiciones están correlacionadas, si cada combinación de sus valores no puede ser especificada en forma independiente.

Cuando se encuentra una combinación de condiciones que no pueden existir simultáneamente, también existe una correlación.

En el ejemplo anterior se tienen las siguientes relaciones:

<i>INDEPENDIENTES</i>	<i>DEPENDIENTES</i>	<i>CORRELACIONADAS</i>
AAA BBB CCC	XXX	A, C

A  
B  
C  
D

Las variables de entrada AAA, BBB, CCC y las condiciones A,B, C y D son independientes. La variable XXX es dependiente. También las condiciones A y C están correlacionadas porque la combinación AC es imposible. Un número igual a dos que no es mayor que uno es imposible.

Se debe exponer la lista de múltiples condiciones incluyendo las combinaciones posibles de las correlacionadas:

AC - AAA > 1     AAA=2  
 $\bar{A}\bar{C}$  - AAA > 1     AAA NOT=2  
 $\bar{A}C$  - AAA <= 1     AAA NOT=2

Los resultados de este tipo de análisis son usados para identificar caminos de prueba imposibles, así que la persona que analiza puede verificar que son imposibles y que no debe perderse tiempo tratando de ejecutar estos caminos.

En relación a los valores que puede tomar XXX, o sea las variables dependientes, se debe listar la combinación de declaraciones, las cuales cuando se ejecutan cambian el valor de la variable dependiente.

1 2     declaración 1 y 2 se ejecutan.  
 $1 \bar{2}$      declaración 1 es la única ejecutada.  
 $\bar{1} \bar{2}$      ninguna declaración se ejecuta.  
 $\bar{1} 2$      declaración 2 es la única ejecutada.

---

Esto significa que cuando se elige el camino, las declaraciones 1 y 2 son ejecutadas, lo que indica por lo tanto que  $XXX = CCC/AAA$  y entonces  $XXX = CCC + 1$ .

Se toma el segundo camino, se ejecuta la declaración 1 y se brinca la segunda, lo que indica que  $XXX = CCC/AAA$ .

### 3) Cobertura de una Ruta o Camino.

Usando las técnicas basadas en la lógica, se definen los caminos que a través del programa serán probados. Al principio podría parecer que se quiere probar todos los caminos de una unidad de prueba. En una unidad o programa sin ciclos podría ser deseable, pero no en unidades con ciclos ya que esto podría ser impráctico.

Supóngase que hay un programa con tres ciclos y que puede ejecutarse hasta 10 veces. El número de caminos únicos que hay en el programa es de 88,572, en base a que es:

$$\begin{array}{cccccccccccc} 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 = 88,572 \end{array}$$

Si el ciclo se ejecuta 20 veces, hay 5,230,176,600 caminos únicos en la unidad.

La prueba exhaustiva de todos los caminos de un programa no es práctico. El objetivo del analista que hace la prueba, es definir un grupo razonable de caminos que cuando se ejecuten tendrán alta posibilidad de descubrir la mayoría de los defectos.

### 4) Complejidad.

El número o nivel de complejidad puede usarse para evaluar la complejidad de un módulo o para determinar el número mínimo de rutas que deberían ejecutarse durante la prueba.

El número de complejidad está relacionado con la lógica de una unidad más que con los recursos de computación requeridos para ejecutarla, o su tamaño. Mientras más alto sea el número más compleja será.

El tamaño de la unidad determinado por el número de líneas de código, no es una buena que indique qué tan mantenible o susceptible de pruebas es una unidad.

No es conveniente desarrollar unidades de alta complejidad. Y por ello, los proyectos pueden limitar la complejidad de los programas, de tal manera que sean más fáciles de mantener, probar, reutilizar y de comprender. Un número de complejidad igual a 10 es el sugerido como límite de un programa.

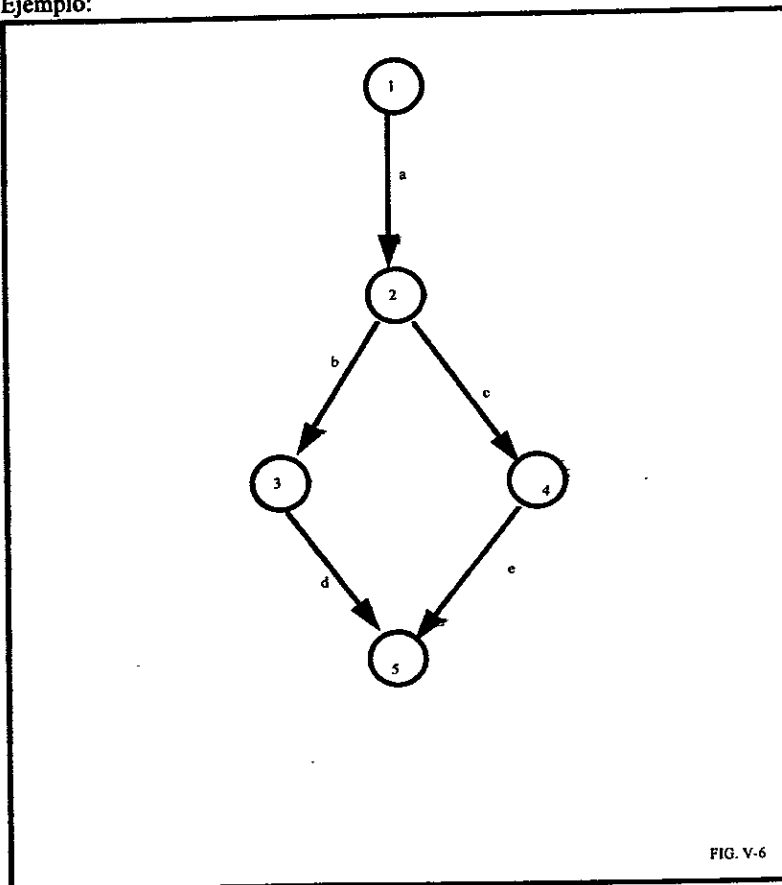
Un número de complejidad 7 es el recomendado para el pseudocódigo, pues módulos muy complejos tienden a tener más defectos, así que limitar la complejidad incrementa la facilidad de probar las unidades.

### Métodos para calcular la complejidad

Existen tres métodos para calcular la complejidad de una unidad, pero primero se debe crear el diagrama de flujo respectivo.

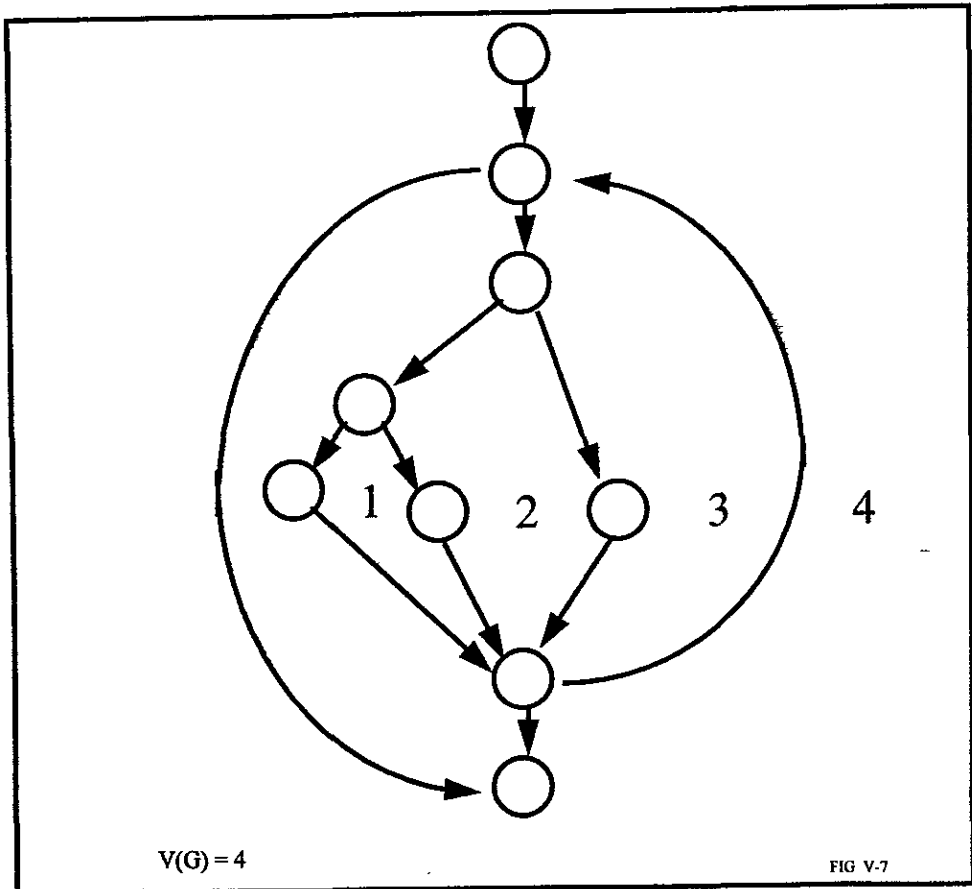
1) Se cuentan el número de nodos (N) y el número de flechas (E) y se aplica la fórmula,  $V(G) = E - N + 2$ , donde  $V(G)$  es el número de complejidad.

Ejemplo:



La complejidad es de 2.

2) Una región es el espacio enmarcado por las flechas y los nodos. Este método asume que las flechas no cruzan una a otra. Toda el área de afuera también es considerada como una región.  
Ejemplo:



3) Cada nodo de decisión, es decir: aquel que implique tomar un camino, contribuye en  $n-1$  a la complejidad. Por ejemplo, 3 nodos de decisión contribuyen a  $3-1$  al número de complejidad.

Así que el número de condiciones +1 determina la complejidad de la unidad: Si se tienen 3 nodos de decisión, cada uno contribuye n-1 por la decisión que se toma:

nodo 1 tiene 2 caminos	entonces $2-1 = 1$
nodo 2 tiene 2 caminos	entonces $2-1 = 1$
nodo 3 tiene 3 caminos	entonces $2-1 = 1$

total de condiciones 3

total de condiciones + 1 = Número de Complejidad  
 3 + 1 = 4

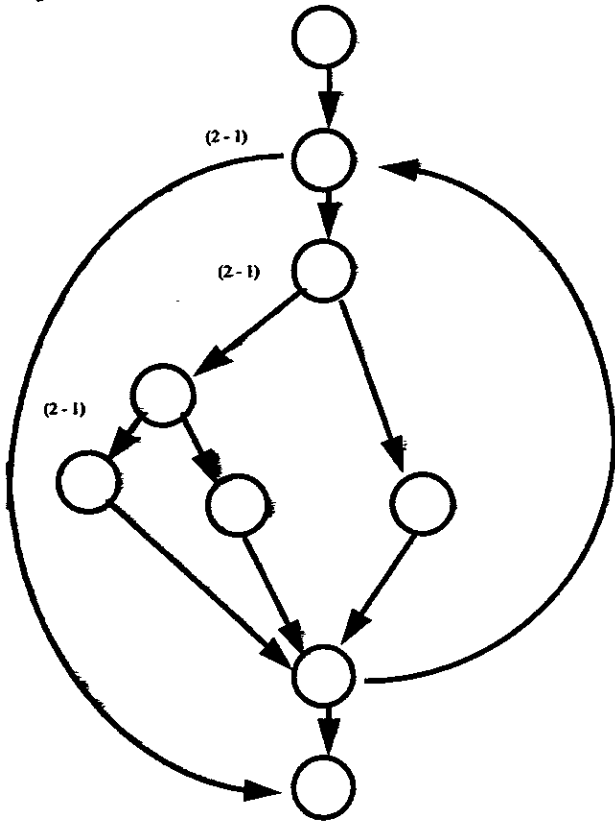


FIG V-8



Lo agradable de este método es que se puede evitar el diagrama de flujo, y se puede ir al código del programa, contar el número de condiciones que tiene y sumarle 1, lo que dará el número de complejidad.

### **Determinación de caminos para la cobertura.**

#### **1) Crear los diagramas de flujo.**

Se deben crear los diagramas desde el momento de generar el código fuente.

Las etiquetas que se asignen a la gráfica deben ser asignadas de tal manera que provean la mayor información para cuando se hace el análisis.

#### **2) Efectuar el análisis de cobertura apropiado.**

Si el plan de pruebas indica que se va a hacer una cobertura de condición múltiple, se puede hacer lo siguiente:

- Identificar las condiciones.
- Identificar las decisiones compuestas y agregarla a la lista de condiciones.
- Identificar condiciones y variables dependientes o correlacionadas.
- Expandir la lista de condiciones para incluir combinaciones posibles de variables dependientes y condiciones correlacionadas.

#### **3) Calcular el número de complejidad.**

Usar uno de los tres métodos para calcular  $V(G)$ .

#### **4) Determinar los caminos y crear la Tabla de Decisión.**

Cuando se está en proceso de determinar los caminos, se debe tomar en cuenta que:

- La cobertura deseada debe alcanzarse. Si esta es una cobertura de condiciones múltiples, entonces todas las combinaciones de las condiciones deben ser tomadas en ambos valores válidos.
- Cada flecha en el diagrama de flujo debe ser atravesada al menos una vez.
- Se debe hacer por lo menos la cobertura que menciona el Número de complejidad  $V(G)$  en la unidad probada.

Esto significa que si la complejidad de la unidad es 8, entonces debe haber por lo menos 8 caminos en la unidad. Se pueden tener más de 8 caminos.

Se debe ser sistemático cuando se eligen los caminos. Se debe iniciar eligiendo un camino base, que ejecute las funciones principales en lugar de los procesos de condición de error. Después se debe usar una de las siguientes técnicas para seleccionar el resto de los caminos:

- Tomar la última decisión en el camino y trabajar hacia arriba, siguiendo las decisiones.
- Tomar la primera decisión y trabajar hacia abajo, siguiendo las decisiones.
- Elegir caminos de izquierda a derecha o viceversa cruzando el diagrama.

Ahora se debe crear la tabla de decisiones en base a las condiciones y las flechas. Listar los identificadores de casos de prueba en la parte de arriba. Conforme se va definiendo cada camino de prueba, se deberá ir llenando la tabla de decisión.

Se debe usar un 1 para indicar que la condición es verdadera o que se cruza una flecha, un 0 para indicar que la combinación es falsa, y un blanco para indicar que la flecha no se cruza.

Se debe continuar definiendo casos de prueba hasta que lo siguiente ocurra:

- Cada fila en la sección de condiciones tiene al menos un "1" y al menos un cero. (La sección de condiciones son las filas de arriba de la tabla).
- Cada fila en la sección de flechas tiene por lo menos un "1" en cada una. (La sección de flechas son las filas de abajo de la tabla).
- Existen menos tantas columnas en la tabla de decisión como el número de complejidad. (Cada columna es un caso de prueba).

#### 5) Definir casos de Prueba.

La tabla de decisiones se requiere para determinar los casos de prueba. Cada columna define las entradas. Ahora bien, es necesario referirse a los requerimientos y especificaciones para determinar que resultados deben esperarse.

- Todos los casos de prueba deben ser documentados. La documentación ayuda durante el desarrollo a ejecutar las pruebas, en la depuración de errores y en la reelección de los casos de prueba.
- Las pruebas no deben ser improvisadas (evitar la técnica del tiroteo).

- Las pruebas deben ser reproducibles. Esto ayuda cuando se hacen depuraciones de errores, y es especialmente difícil corregirlos cuando no existe una descripción de la falla. Y eso es precisamente un caso de prueba reproducible, una descripción exacta de la falla.
- Los casos de prueba deben poder ser re-ejecutados para verificar que los defectos han sido corregidos. Recuérdese que un programa no está completo hasta que se ha terminado la documentación de los casos de prueba.
- Se debe incluir una descripción de los resultados esperados, que ayude a descubrir errores. Algunas ocasiones la persona que ejecuta las pruebas, no nota las fallas. Así que pruebas bien documentadas incluyen una descripción de los resultados que se esperan cuando se ejecutan las pruebas. De esta manera se puede verificar que los objetivos de la prueba se han logrado.

### **Elementos de la documentación de casos de prueba.**

La documentación de los casos de prueba debe contener los siguientes elementos:

- Un identificador único por caso de prueba.
- El objetivo del caso de prueba indicando el criterio que se está utilizando en la prueba, además de identificar los planes de prueba que soportan estos casos.
- Especificaciones de las entradas. Se enlistan las entradas requeridas para ejecutar el caso de prueba. Estos pueden ser datos actuales o una referencia a un archivo con ciertos datos.
- Los resultados esperados. los defectos pueden ser claramente visibles durante la ejecución de la prueba.
- Procedimientos de ejecución. Proporcionar en forma detallada paso a paso, las instrucciones de como ejecutar un caso de prueba. Se debe incluir información, acerca de cómo prepararlo, iniciarlo, procesarlo, detenerlo y reiniciarlo. Indicar cómo se pueden evaluar los resultados, necesidades especiales de equipo y programación, así como el ambiente necesario deben ser totalmente descritos.

Las siguientes páginas muestran algunos ejemplos para la documentación.

**MUESTRA DE LA FORMA PARA EL EXAMEN DE LA DESCRIPCIÓN DE UN CASO**

**Identificador:** \_\_\_\_\_

**Planes de Prueba:** \_\_\_\_\_

**Prueba de acciones validadas:** \_\_\_\_\_

**Prueba de acciones no validadas:** \_\_\_\_\_

**Descripción de la**

**prueba:** \_\_\_\_\_

**Objetivos:** \_\_\_\_\_

**Procedimiento:** \_\_\_\_\_

**Especificaciones de entrada o referencias:** \_\_\_\_\_

**Resultados esperados o referencias:** \_\_\_\_\_

**Status: Aprobado/Reprobado/Reexaminar.**

## **CAPÍTULO VI**

### **CASO PRÁCTICO DE APLICACIÓN DE TÉCNICAS DE PRUEBA**

#### **Descripción del plan del proyecto para el sistema de compra de Vehículos.**

El sistema de compra de vehículos contiene las siguientes funciones:

Define el vehículo a ser comprado

Calcula el costo de manufactura, compras locales, compras de importación y el total del costo del vehículo.

Verifica el programa de producción del material.

Crea una orden de venta del vehículo.

Transmite la orden de material al proveedor.

Proporciona un recibo de orden al cliente.

El sistema propuesto se regirá por la metodología definida a lo largo de los capítulos 1 al 5:

- análisis.
- diseño de negocios.
- diseño técnico.
- construcción.
- PRUEBAS.
- Implantación.
- soporte a producción.

El proyecto tomará 12 meses partiendo del inicio de la fase de negocios hasta la aceptación formal del usuario, durante la fase de implantación.

El staff de programación consta de 5 a 10 Ingenieros de Sistemas. La mitad de ellos tiene un promedio de 1 a 2 años de experiencia en este tipo de aplicaciones. El resto de los Ingenieros tiene de 3 a 9 meses de experiencia en codificación.

### DIAGRAMA DE CONTEXTO

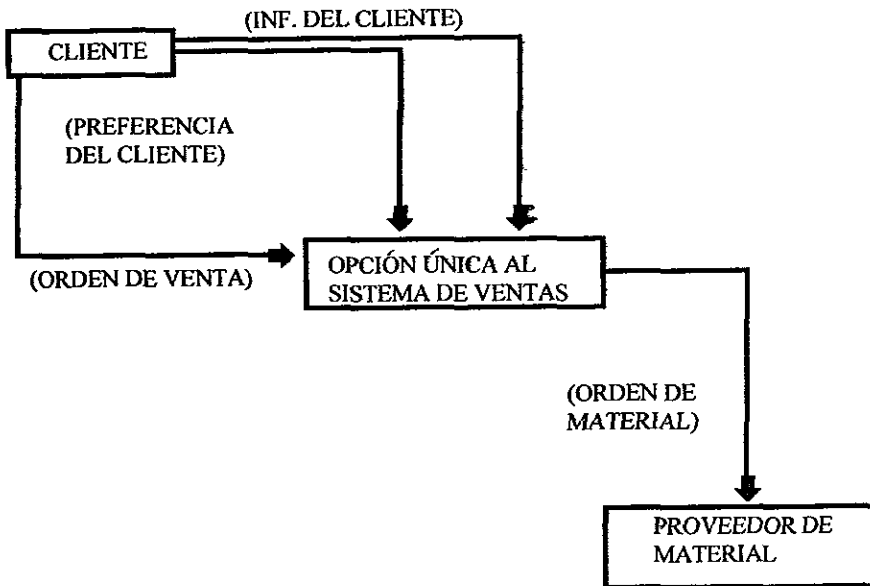


FIG. VI-1

### **Lista de eventos para el sistema de compra de vehículos.**

- El vendedor introduce la información del cliente.
- El cliente decide el vehículo que quiere comprar.
- El cliente y el vendedor deciden las opciones que el vehículo llevará.
- El sistema calcula el costo de manufactura, compras locales, las compras de importación y el costo total del vehículo.
- Los programas de producción son verificados para cada opción y modelo.
- La orden de venta es introducida al sistema con la información del vehículo, del distribuidor y del cliente.
- La orden del vehículo es transmitida al sistema de producción para ser procesada.
- Se entrega una copia al cliente.

I.- A partir de la siguiente descripción de la *OPCION 1* del proceso de ventas, se debe desarrollar un diagrama de causa y efecto para el segundo paso en el proceso. (La descripción del paso dos inicia en la página siguiente).

II.- Se deberá usar la gráfica desarrollada para crear una tabla de decisión.

III.- Identificar los casos de prueba iniciales necesarios para el segundo paso del proceso.

### **Descripción de la opción 1 del sistema de ventas.**

El sistema ayudará a completar el segundo paso del proceso de venta involucrado en la colocación de una orden de venta de un vehículo. La segunda opción calcula el precio de varios paquetes de opciones.

El primero y tercer paso será manual, pero están documentados para dar una figura completa de la opción 1.

El primer paso es la verificación del cliente como alguien elegible para comprar un vehículo bajo un plan de descuento.

---

El segundo paso, el primero a ser automatizado, es ejecutado cuando un cliente se ha decidido por algún modelo en particular y desea comprarlo. El vendedor utilizando una terminal y en forma interactiva, describe al vehículo en términos de opciones a ser especificadas. En otras palabras, el usuario tendrá la posibilidad de agregar y eliminar opciones para la manufactura del auto.

El precio del vehículo está compuesto por los gastos de manufactura y de los proveedores. Estos son obtenidos de los archivos internos.

Si un modelo, una opción o una acción es inválida, o se hace un intento de borrar una opción, un mensaje de error será desplegado.

El punto final, es cuando el vehículo es ordenado exactamente como el cliente lo quiere. Entonces se produce la orden en papel y el material es ordenado al proveedor, y se genera la orden de producción a la planta de ensamble.



DIAGRAMA DE ESTRUCTURA

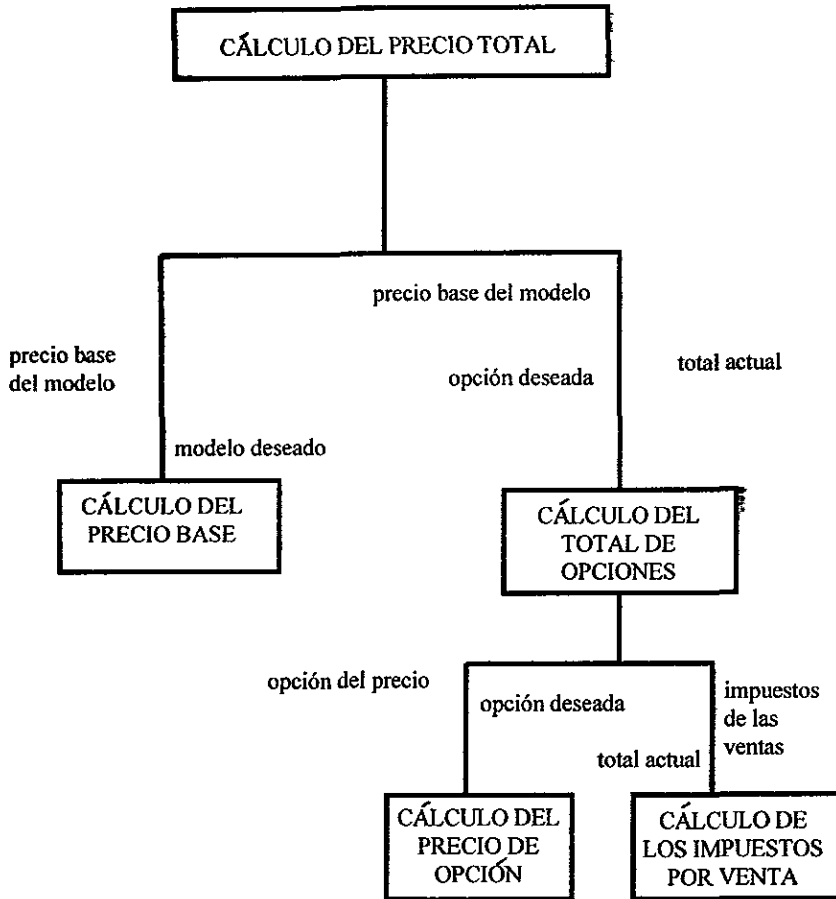


FIG. VI-2

## DEFINICIÓN DE CAUSAS Y EFECTOS

### CAUSAS

1) ACCIÓN DE ADICIÓN RECIBIDA.

2) MODELO VÁLIDO.

3) OPCIÓN VÁLIDA.

4) PORCENTAJE DE DESCUENTO.

5) ACCIÓN DE BORRADO RECIBIDO.

6) ADICIÓN PREVIA

### EFECTOS

91) ADICIÓN CÁLCULO Y MENSAJE.

92) ERROR DE ACCIÓN NO VÁLIDA.

93) ERROR DE ENTRADA NO VÁLIDA.

94) BORRADO, CÁLCULO Y MENSAJE.

95) ERROR DE BORRADO NO VÁLIDO.

# DIAGRAMA DE CAUSA Y EFECTO (ALTERNATIVA 1)

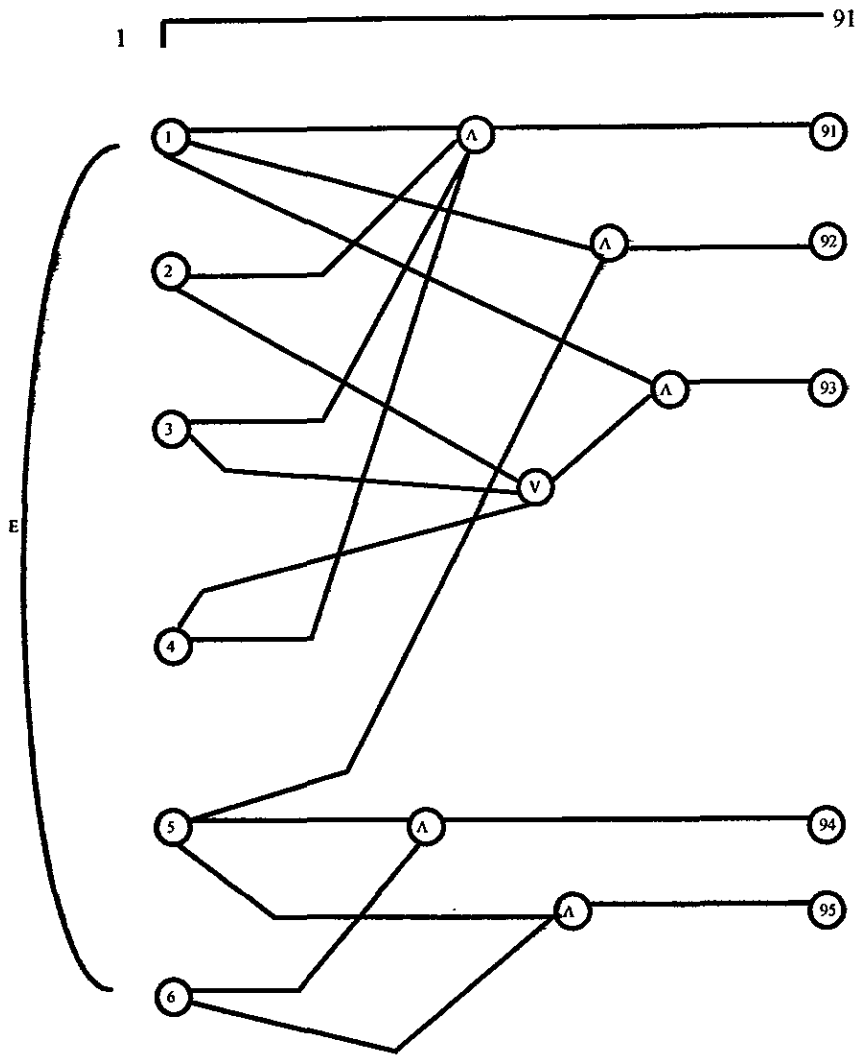


FIG. VI-3

## DIAGRAMA DE CAUSA Y EFECTO (ALTERNATIVA 2)

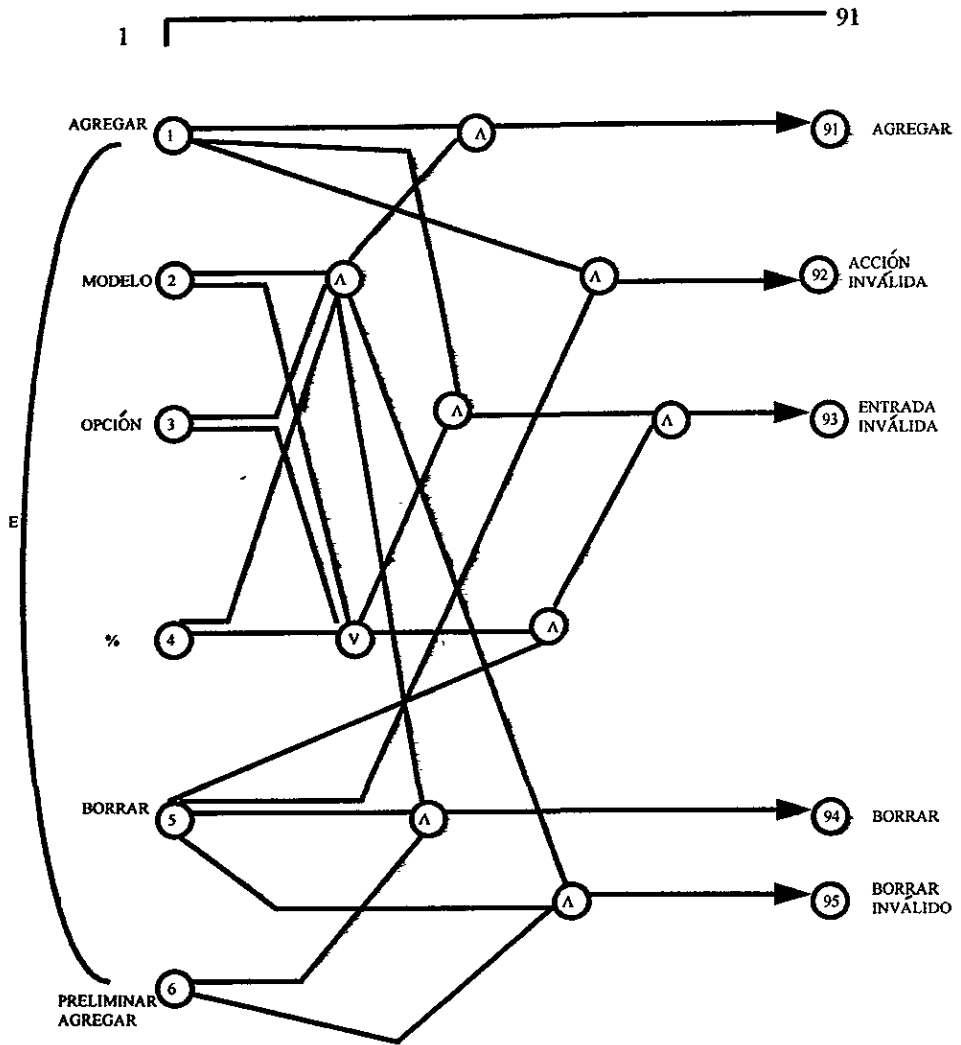


FIG VI-4

**TABLA DE DECISIÓN DE LA ALTERNATIVA 1**

	1	2	3	4	5	6	7
1	1	0	1	1	1	0	0
2	1		0	1	1		
3	1		1	0	1		
4	1		1	1	0		
5	0	0	0	0	0	1	1
6						1	0
91	1	0	0	0	0	0	0
92	0	1	0	0	0	0	0
93	0	0	1	1	1	0	0
94	0	0	0	0	0	1	0
95	0	0	0	0	0	0	1

**TABLA DE DECISIÓN DE LA ALTERNATIVA 2**

	1	2	3	4	5	6	7	8	9	10
1	1	0	1	1	1	0	0	0	0	0
2	1		0	1	1	0	1	1	1	1
3	1		1	0	1	1	0	1	1	1
4	1		1	1	0	1	1	0	1	1
5	0	0	0	0	0	1	1	1	1	1
6									1	0
91	1	0	0	0	0	0	0	0	0	0
92	0	1	0	0	0	0	0	0	0	0
93	0	0	1	1	1	1	1	1	0	0
94	0	0	0	0	0	0	0	0	1	0
95	0	0	0	0	0	0	0	0	0	1

## **CASOS DE PRUEBA DERIVADOS DE LA TABLA DE DECISIÓN DE CAUSA Y EFECTO (ALTERNATIVA I)**

1) Entrada: ACCIÓN DE ADICIÓN.  
MODELO VÁLIDO DEL VEHÍCULO  
OPCIÓN VÁLIDA  
PORCENTAJE DE DESCUENTO VÁLIDO  
Salida Esperada: LA OPCIÓN ADICIONADA, EL PRECIO CALCULADO Y DESPLEGADO.

2) Entrada: ACCIÓN INVÁLIDA.  
Salida Esperada: MENSAJE DE ACCIÓN INVÁLIDA

3) Entrada: ACCIÓN DE ADICIÓN  
MODELO INVÁLIDO DEL VEHÍCULO  
OPCIÓN VÁLIDA  
PORCENTAJE DE DESCUENTO VÁLIDO.  
Salida esperada: MENSAJE DE ENTRADA INVÁLIDA.

4) Entrada: ACCIÓN DE ADICIÓN  
MODELO VÁLIDO DEL VEHÍCULO  
OPCIÓN INVÁLIDA  
PORCENTAJE DE DESCUENTO VÁLIDO  
Salida esperada: MENSAJE DE ENTRADA INVÁLIDA.

5) Entrada: ACCIÓN DE ADICIÓN.  
MODELO VÁLIDO DEL VEHÍCULO  
OPCIÓN VÁLIDA  
PORCENTAJE DE DESCUENTO INVÁLIDO  
Salida esperada: MENSAJE DE ENTRADA INVÁLIDA.

6) Entrada: ACCIÓN DE BORRADO.  
OPCIÓN A SER BORRADA  
OPCIÓN EN LA LISTA DE OPCIONES  
Salida esperada: OPCIÓN ES BORRADA, EL PRECIO CALCULADO Y DESPLEGADO.

7) Entrada: ACCIÓN DE BORRADO  
OPCIÓN A SER BORRADA  
OPCIÓN NO ESTÁ EN LA LISTA DE OPCIONES  
Salida esperada: MENSAJE DE BORRADO INVÁLIDO.

## **PARTICIÓN DE EQUIVALENCIAS Y ANÁLISIS DE VALORES DE FRONTERAS**

### **1) PORCENTAJE DE DESCUENTO PARA EL MODELO.**

Clase Válida	-	Cualquier valor entre 15 y 20 inclusive.
Clase Inválida	-	Cualquier valor arriba de 20.
Clase Inválida	-	Cualquier valor abajo de 15.
Clase Inválida	-	Cualquier valor no numérico.

### **VALORES DE FRONTERA:**

Válido	-	15, 20.
Inválido	-	14, 21.
Inválido	-	A.

### **2) ACCIÓN SOBRE LA OPCIÓN**

Clase Válida	-	acción de Adición.
Clase Válida	-	acción de Borrado.
Clase Inválida	-	cualquier otra acción.

### **3) MODELO DEL VEHÍCULO:**

Clase válida	-	cualquier vehículo con descuento.
Clase Inválida	-	cualquier otro vehículo.

### **4) OPCIONES:**

Clase válida	-	cualquier opción disponible para este vehículo.
Clase Inválida	-	cualquier otra opción.

**DETERMINACIÓN DE LAS CONDICIONES QUE OCURREN EN UN PSEUDO  
CÓDIGO DE UNA DE LAS FUNCIONES PRINCIPALES DEL SISTEMA DE COMPRA  
DE VEHÍCULOS.**

PROCESO NARRATIVO  
PROCESO : 2.2  
NOMBRE DEL PROCESO : CÁLCULO DEL PRECIO DE LAS OPCIONES

SET CURRENT\_TOTAL TO MODEL\_BASE\_PRICE

GET DESIRED\_OPTION FROM THE SCREEN

DO WHILE DESIRED\_OPTION IS NOT BLANK

SET THE VALID\_ACTION\_FLAG TO TRUE

IF DESIRED\_OPTION IS VALID.ON THIS CAR

GET THE ACTION FOR DESIRED\_OPTION FROM THE SCREEN

IF THE ACTION IS ADD

DO DETERMINE\_OPTION\_PRICE

ADD OPTION\_PRICE TO CURRENT\_TOTAL GIVING CURRENT\_TOTAL

PUT THE DESIRED\_OPTION INTO THE OPTION\_LIST

ELSE

IF THE ACTION IS DELETE

IF DESIRED\_OPTION IS IN THE OPTION\_LIST

DO DETERMINE\_OPTION\_PRICE

SUBSTRACT\_OPTION\_PRICE FROM CURRENT\_TOTAL GIVING

CURRENT\_TOTAL

TAKE THE DESIRED\_OPTION OUT OF THE OPTION\_LIST

ELSE

SET THE VALID\_OPTION\_FLAG TO FALSE

PUT OPTION NOT IN OPTION\_LIST MESSAGE TO THE SCREEN

ENDIF

ELSE

SET THE VALID-ACTION\_FLAG TO FALSE

PUT BAD ACTION ENTERED MESSAGE TO THE SCREEN

ENDIF

ENDIF



IF THE VALID\_ACTION FLAG IS TRUE  
DO CALCULATE SALES\_TAX  
ADD SALES\_TAX TO CURRENT\_TOTAL GIVING CURRENT\_COST  
PUT CURRENT\_COST TO THE SCREEN

ENDIF

ELSE

    PUT BAD OPTION ENTERED MESSAGE TO THE SCREEN

ENDIF

    GET DESIRED\_OPTION FROM THE SCREEN

END DO WHILE

MOVE CURRENT\_COST TO CURRENT\_TOTAL



DIAGRAMA DE FLUJO SIMPLIFICADO

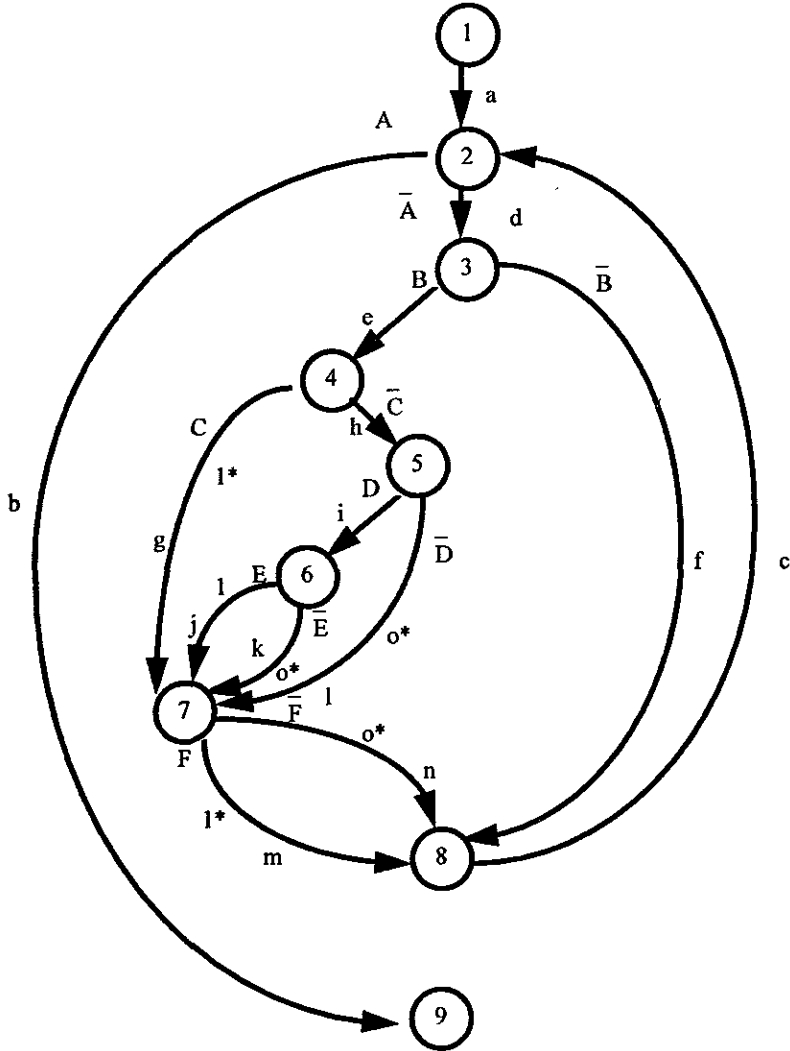


FIG VI-6

## ETIQUETA EN LA GRÁFICA

## CONDICIONES

A	LA OPCIÓN DESEADA ESTÁ EN BLANCO
B	LA OPCIÓN DESEADA ES VÁLIDA EN EL AUTO
C	LA ACCIÓN ES ADICIONAR
D	LA ACCIÓN ES BORRAR
E	LA OPCIÓN ESTÁ EN LA LISTA DE OPCIONES
F	LA SEÑAL DE ACCIÓN VÁLIDA ES ACTIVADA

## CONDICIONES CORRELACIONADAS

Las condiciones C y F son correlacionadas.  
Las condiciones D y F son correlacionadas.  
Las condiciones E y F son correlacionadas.

## RUTAS IMPOSIBLES DE SEGUIR EN BASE A LAS CONDICIONES CORRELACIONADAS

$\overline{CF}$

$\overline{DF}$

$\overline{EF}$

$\overline{EF}$

## PROCESOS DEPENDIENTES

La bandera `VALID_ACTION_FLAG` es una variable dependiente.

Debido a que la ejecución de la instrucción `IF ("IF THE VALID ACTION FLAG IS TRUE")` depende del valor que contenga `VALID_ACTION_FLAG`, la condición (F) es un proceso dependiente.

Aplicando cualquier método para calcular la complejidad del diagrama, se obtiene una complejidad igual a 7.

En base a ello se puede obtener la siguiente tabla de decisión que muestra otros posibles casos de prueba que pudieran ser necesarios para este proceso.

	1	2	3	4	5	6	7
A	0	0	0	0	0	1	
B	1	1	1	1	0		
C	1	0	0	0			
D		1	1	0			
E		1	0				
F	1	1	0	0			
a	1	1	1	1	1	1	
b						1	
c	1	1	1	1	1		
d	1	1	1	1	1		
e	1	1	1	1			
f					1		
g	1						
h		1	1	1			
i		1	1				
j		1					
k			1				
l				1			
m	1	1					
n			1	1			

Solo seis casos son posibles. El pseudo código puede ser reestructurado para reducir la complejidad a 6.

## CASOS DE PRUEBA POSIBLES

1) Entrada: OPCIÓN VÁLIDA  
ACCIÓN DE ADICIÓN

Salida esperada: SUMA EL PRECIO AL TOTAL  
SUMA LA OPCIÓN A LA LISTA

2) Entrada: OPCIÓN VÁLIDA  
ACCIÓN DE BORRAR UNA OPCIÓN ADICIONADA PREVIAMENTE

Salida esperada: RESTA EL PRECIO DEL TOTAL  
BORRA LA OPCIÓN DE LA LISTA

3) Entrada: OPCIÓN VÁLIDA  
ACCIÓN DE BORRAR UNA OPCIÓN NO ADICIONADA PREVIAMENTE

Salida esperada: DESPLEGAR EL MENSAJE DE QUE LA OPCIÓN NO ESTÁ INCLUIDA EN  
LA LISTA

4) Entrada: OPCIÓN VÁLIDA  
ACCIÓN INVÁLIDA

Salida esperada: DESPLIEGA MENSAJE DE QUE UNA ACCION ERRONEA SE INTRODUJO

5) Entrada: OPCIÓN INVÁLIDA

Salida esperada: DESPLIEGA MENSAJE DE QUE UNA ACCIÓN ERRÓNEA SE INTRODUJO

6) Entrada: OPCIÓN EN BLANCO

Salida esperada: TERMINAR EL PROCESO

## CAPÍTULO VII

### RESULTADOS

Esta tesis documenta una parte representativa de la aplicación de las técnicas de prueba en un sistema de información, en donde se pueden apreciar los beneficios de algo planeado, con uso de un método y disciplina para producir mejores resultados.

Sin embargo, en este capítulo final quiero ampliar y proporcionar mayor detalle de los resultados obtenidos al aplicar diversas partes de la metodología planteada, en diferentes proyectos de implementación de sistemas automatizados.

La idea de crear esta metodología surge de la frecuente experiencia de participar en proyectos ya iniciados e incluso instrumentados, que estuvieron llenos de cambios, contratiempos, retrasos, altos costos y frustraciones tanto de las áreas de sistemas como de los usuarios o clientes finales, en los que gracias al talento humano llegaron a buen final pero con un alto costo moral y financiero. Así que buscando crear sistemas con mayor calidad, menos tropiezos y mejor aceptación del usuario, busqué la manera de crear algo que no fuera tan dependiente de la experiencia del programador, del líder de proyecto e incluso del usuario, y que funcionara como una guía que permitiera considerar y cuidar los aspectos más importantes al momento de crear, diseñar, construir e instrumentar sistemas de información. Durante experiencias pasadas identifiqué que la parte más endeble era la correcta herramienta o guía para aplicar pruebas de funcionalidad y que estas eran diseñadas en base al criterio y experiencia de los involucrados, mas que como un método de costo-beneficio que ayudara a asegurar los mejores resultados.

Cada uno de los aspectos incluidos en esta propuesta fueron obtenidos de experiencias reales que le dieron forma y definición, misma que ahora describo en forma resumida:

La organización en la que hoy participo es una institución privada cuyo principal aspecto de negocio es el desarrollo de sistemas de cómputo. A lo largo de más de 2 años observé y experimenté las situaciones descritas al inicio de este capítulo, y conforme se fueron dando las circunstancias se fue creando esta metodología.

La metodología fue aplicada con la tesis de que si se tenían guías y herramientas, así como un método que resaltaran los aspectos a cuidar al instrumentar sistemas, estos podrían generarse con menores retrasos en su entrega final, dentro del presupuesto planeado y sobre todo ser aceptados por sus usuarios finales.

La organización aceptó aplicar y monitorear la metodología en una serie de sistemas que conformaban el plan estratégico y de negocios de un cliente, el cual incluía múltiples aplicaciones administrativas.

Los grupos de desarrollo eran variados tanto en experiencia como en el número de integrantes, así que la aplicación consistió en utilizar estas técnicas y herramientas según la fase o etapa en que cada proyecto se encontraba.

En total fueron 12 proyectos que a lo largo de 2 años se consideraron y que se listan a continuación:

1. Sistema de Nómina con plataforma Cliente-Servidor y base de Datos relacional.
2. Sistema de control de tripulaciones.
3. Sistema de asignación automática de tripulaciones.
4. Sistema de Ingeniería para control de Peso y Balance.
5. Sistema de información y Reservas y publicidad a través de Internet.
6. Sistema de Automatización de operaciones de Agentes de Ventas.
7. Sistema de Viajero frecuente.
8. Sistema de Reconocimiento a pasajeros.
9. Sistema de control de equipaje perdido.
10. Sistema de atención a clientes.
11. Sistema de información gerencial.
12. Sistema de ingresos y determinación del Ingreso y rentabilidad.

Se creó un grupo denominado de "Sistemas Internos" constituido por 5 integrantes, cuya responsabilidad principal era capacitar en el uso de la metodología, apoyar a los grupos en la aplicación y diseño de casos de prueba y documentar los resultados y experiencias. Esto se conserva como material propietario de la compañía y por ello no se presenta en este trabajo de tesis.



Se integró un plan de trabajo único que reflejaba las fechas de inicio y terminación de cada uno y se destacaron los principales módulos de cada sistema. Se documentó lo que se esperaba como resultados y como se verificarían al final de la construcción del sistema. Y antes de proceder a su inicio se tomaron las siguientes acciones:

- Se aplicó una encuesta de satisfacción al cliente que evaluaba los resultados obtenidos por el grupo de sistemas hasta esa fecha, hace 2 años, desde el punto de vista de los usuarios finales, que en su mayoría eran los mismos que habían participado en proyectos anteriores y que ahora también lo harían en los nuevos proyectos. El resultado fue de 3.5 en la escala de 1 a 7.
- Se aplicó también una encuesta de satisfacción de los empleados que medía el grado de satisfacción de cada uno de los que formaron parte del equipo de sistemas, en relación a su trabajo y funciones actuales, trabajo en equipo, sus compañeros, clientes y clima organizacional. El resultado fue de 3.2 en la escala de 1 a 7.
- Se acordó sesionar semanalmente para evaluar el avance, resultados y problemas encontrados durante este proceso.
- Se designaron líderes de proyecto de cada área y se realizaron presentaciones tanto a los usuarios finales como al personal de sistemas.

Y así inicio un trabajo de aplicación, prueba, monitoreo y documentación de resultados que duró el tiempo suficiente para llegar al vencimiento de la fecha de terminación de cada proyecto; se aplicó nuevamente una encuesta de satisfacción a los clientes y a los empleados y los resultados se presentan a continuación en forma resumida:

- 10 de 12 proyectos fueron concluidos en tiempo.
- En los 12 proyectos participó el usuario final en forma muy amplia.
- 8 proyectos fueron concluidos con el presupuesto originalmente previsto y los 4 restantes con una variación excedente dentro del 5%.
- Todos los sistemas y procedimientos anteriores fueron migrados hacia las nuevas plataformas y servicios.
- Los manuales y procedimientos se editaron una sola vez y después de 6 meses no han presentado correcciones.

- Los 12 sistemas están siendo utilizados por los usuarios.
- El tiempo máximo de operación en paralelo de los sistemas anteriores y nuevos fue de solo 1 mes.
- El resultado de la encuesta de satisfacción de los usuarios finales fue de 5.2 en la escala de 1 a 7, lo que representó un crecimiento muy positivo superior al 40% lo que es muy satisfactorio considerando las evaluaciones de los últimos 5 años.

Los principales elementos de valor que cada uno de los usuarios retroalimentó se sintetizan a continuación:

1. *“El área de sistemas entendió mis requerimientos y canalizó adecuadamente mis expectativas”*
2. *“Mis proyectos salieron a tiempo y sin mayores contratiempos”*
3. *“Estuve realmente involucrado en mi sistema”*
4. *“Por primera vez lo que dice el manual coincide con lo que el sistema hace”*
5. *“El mismo grupo de desarrollo debe continuar con todos los proyectos”*
6. *“El proceso fue pesado y tedioso en algunas ocasiones, pero los resultados son muy buenos”*
7. *“El área de sistemas debería tener metodologías para ayudar a los usuarios a definir mejor sus requerimientos y necesidades”*
8. *“El sistema no me falló al momento de utilizarlo por primera vez”.*

- El resultado de la encuesta de satisfacción de los empleados creció significativamente, casi un 50%, de 3.2 a 4.8 en la misma escala de 1 a 7. Los principales elementos de retroalimentación de los empleados se sintetizan a continuación:
  1. *“Los empleados opinaron que se sentían satisfechos con su empleo y las funciones desempeñadas”*
  2. *“Se sintieron satisfechos y sin frustraciones al momento de liberar los sistemas al usuario”*
  3. *“Indicaron que se establecieron reglas y roles justos y se lograron lazos emocionales más fuertes entre los integrantes del grupo”*
  4. *“Sugirieron que la metodología es buena y funcional pero que podría ser más simple para algunos proyectos menores”*
  5. *“Se siente muy sólidos para poder operar y mantener los sistemas”*
  6. *“Consideran que la organización debería compensarlos mejor al lograr mejores resultados y conocimientos en relación a sus compañeros”*
  7. *“Sugieren que el grupo se mantenga conformado de la misma manera para proyectos futuros”*
  8. *“No hubo tantas quejas, conflictos o reclamos con los usuarios. Todo estaba documentado y pudimos tener argumentos para negociar y llegar a acuerdos con los usuarios finales.”*

Conforme a los elementos de medición y retroalimentación, se puede concluir que los resultados obtenidos son los siguientes:

- Más del 80% de los sistemas fueron liberados a tiempo.
- Más del 70% de los proyectos fueron concluidos con el presupuesto original.
- El usuario aceptó la operación y uso de los nuevos proyectos.
- Los sistemas no presentan fallas de funcionalidad en su operación.
- El nivel de satisfacción tanto de los usuarios como de los empleados mejoró y creció ampliamente con porcentaje superior al 40% en términos generales.

En relación al resultado obtenido durante el proceso de aplicación de la metodología , también se obtuvieron experiencias muy positivas que fueron monitoreadas estadísticamente.

A continuación se muestran las gráficas que indican la tendencia positiva de la aplicación de las técnicas de prueba y los casos de prueba desarrollados, y también se muestran algunos casos en los que se puede detectar oportunamente cuando dichos casos de prueba no están siendo exitosos o correctamente aplicados, con el fin de identificar esta situación con oportunidad y evitar un fracaso en el esfuerzo de prueba.

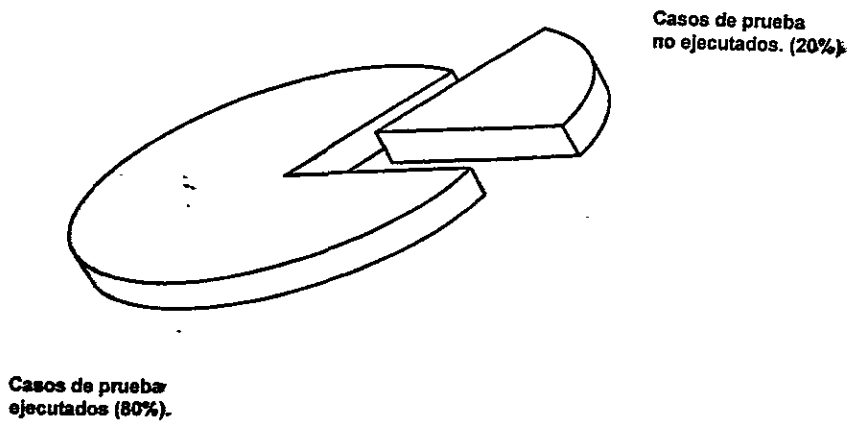
Estas son herramientas que permiten evaluar el esfuerzo de prueba, cómo ha sido su calidad y cuál será el nivel de eficiencia que se obtendrá al final como resultado de la aplicación de los casos de prueba y poder determinar si se debe o no seguir probando.

## ANÁLISIS DE RESULTADOS

1. Considero que los sistemas fueron ampliamente probados de acuerdo al porcentaje alcanzado, que reflejó un 80% y que fue determinado por la siguiente fórmula:

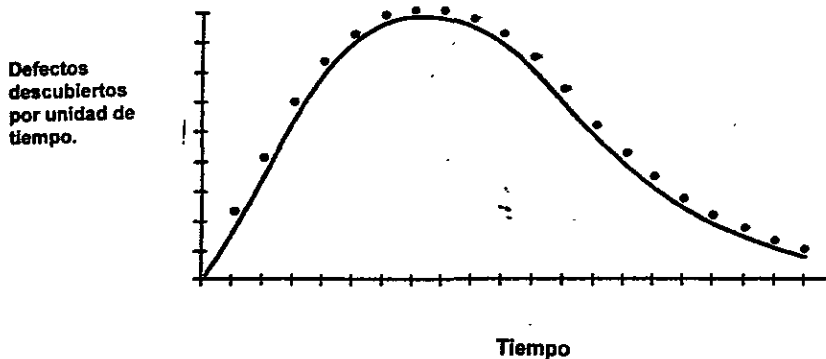
$$\% \text{ del Sistema probado} = \frac{\text{Número de casos de prueba ejecutados} \times 100}{\text{Número de casos de prueba totales}}$$

En cada uno de los proyectos participantes el porcentaje fue mayor al 80%.



2. El segundo análisis de los resultados se basó en la aplicación de la curva de Raleigh, en donde se aplicó la tesis de que conforme se aplicaban los casos de prueba unitarios, la tendencia debería ser hacia descubrir errores y que progresivamente tenderían a aproximarse a cero.

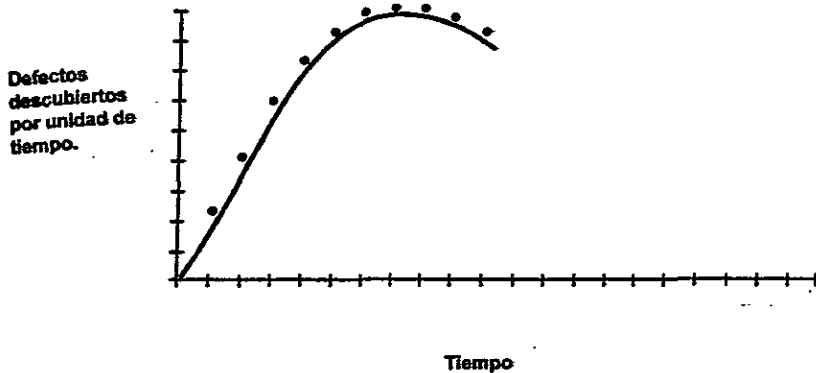
Así que se gráfcaron el número de defectos descubiertos por cada unidad de tiempo durante el proceso de prueba del sistema y de las pruebas de integración, y encontré en general las siguientes tendencias:



Esta gráfica muestra el número de casos de prueba por día. En donde cada uno de los puntos representa la medida actual. Por ejemplo, un punto puede representar 12 defectos descubiertos en el día 7 y otro punto 15 en el día 18.

Cuando el proceso de prueba del sistema y de las pruebas de integración se inicia, el número de defectos detectados es bajo; posteriormente este número se incrementa rápidamente conforme el esfuerzo de prueba avanza. Esto sucede porque todavía existen numerosos defectos en el sistema y es relativamente fácil descubrirlos.

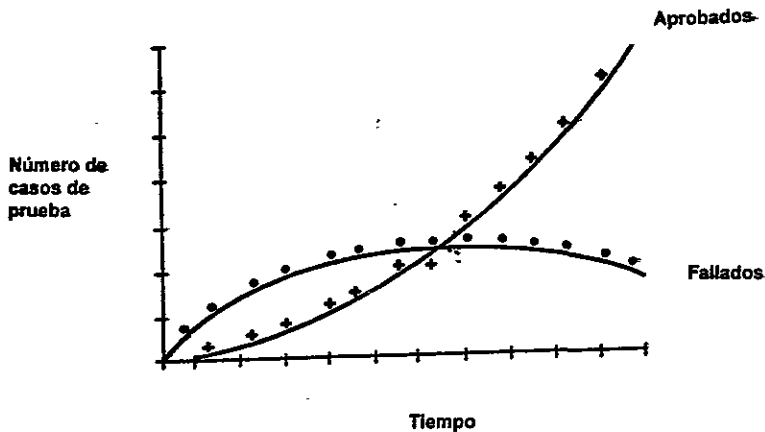
Una de las experiencias que enfrentamos fue la siguiente:



Esta gráfica nos indicó que existía un problema en nuestro esfuerzo de prueba, ya que en vez de que la tendencia fuera hacia cero, ésta iba creciendo. De continuar de esta manera el esfuerzo sería inútil y muy costoso, así que lo mejor fue revisar los casos de prueba aplicados y corregir los problemas identificados.

3. El siguiente análisis de resultado aplicado para evaluar los beneficios de las técnicas de prueba, fue el análisis de casos de prueba exitosos y los fallidos.

En esta técnica se gráfica cada uno de los casos de prueba que fue exitoso, es decir, aquellos generaron los resultados esperados y en forma separada aquellas situaciones en el que el caso de prueba falló.

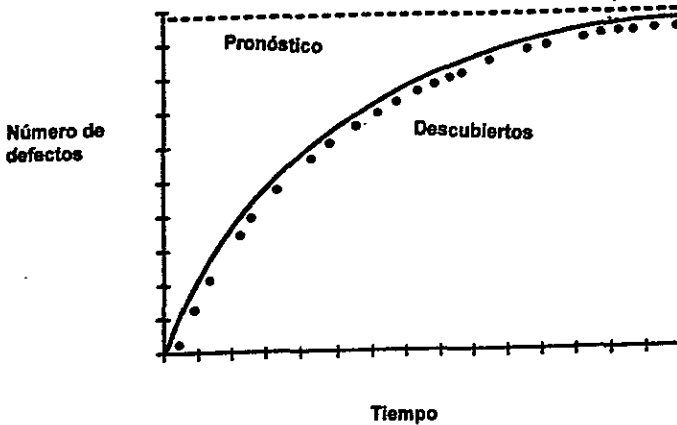


En esta gráfica se puede apreciar como la tendencia debe ser en todos sentidos positiva, es decir, el número de casos exitosos debe ser incremental y decreciendo para los que fallan, lo que es un claro indicativo de que los casos y las técnicas de prueba están funcionando correctamente.



4. El último análisis efectuado fue el histórico, en donde se tomo una media de los sistemas desarrollados anteriormente a este esfuerzo. Esta media representaba el número mínimo de errores encontrados y se definió como el máximo a detectarse utilizando las técnicas de prueba de este trabajo.

Los resultados se muestran a continuación en una gráfica que describe en términos generales el comportamiento de los defectos históricos encontrados en el proceso de prueba.



En todos los análisis efectuados para cada uno de los proyectos, se observó la misma tendencia, lo que lleva a la conclusión de que las técnicas de prueba y los elementos de monitoreo fueron de gran influencia en los resultados obtenidos como resultado final a los clientes y usuario final de cada uno de los proyectos mencionados al inicio de éste capítulo.

## CONCLUSIONES

Después de analizar y evaluar los resultados y haber sido participe de todo el proceso anteriormente descrito, puedo considerar como conclusión que los resultados han sido muy positivos y satisfactorios para cada una de las partes; el usuario final y el área de desarrollo sistemas; que esta metodología fue parte muy importante de ese proceso y que ayudó a mantener disciplina, orden y a proporcionar un método a lo largo del desarrollo de cada uno de los proyectos, y que si bien hubo otros muy valiosos elementos, como capacidad de liderazgo de algunos integrantes, experiencias previas y un amplio deseo de realizar profesionalmente cada uno de los sistemas, la metodología fue un instrumento de gran utilidad para lograr resultados depurados y confiables en un tiempo óptimo, que además incluyó un ingrediente adicional que fue el saber y confiar en que el sistema había sido previamente evaluado y probado y que por lo tanto al presentarse a consideración del usuario final, garantizaba un nivel mínimo de fallas o sorpresas negativas, lo que generó y redujo en un mayor compromiso por parte de los usuarios finales en relación a cada sistema.

Concluyo que la metodología no es en sí misma la clave del éxito de la implantación exitosa de sistemas, ya que alrededor de ella hay muchos otros elementos que participan, pero que sí forma definitivamente parte de la estrategia a seguir para asegurar mejores resultados.

De igual manera puede concluirse que esta metodología propuesta tiene valor al aplicarse, ya que entre sus puntos fundamentales permiten evaluar el nivel de riesgo con que cada sistema se instrumenta o libera a producción; hacer confiables muchos aspectos obvios del desarrollo de sistemas y ofrecer un método para simplificar los programas, depurarlos y asegurar una mejor manera de mantenerlos a futuro. Asimismo esta propuesta presenta flexibilidad al permitir dos niveles de generación de casos de prueba y de actividades de verificación: aquellas que son estrictamente necesarias y consideradas como mínimas y aquellas que son altamente recomendables para incrementar el factor de éxito en cada desarrollo de sistemas, de tal forma que se puede optar por cualquiera de ellas dependiendo del tiempo y presupuesto disponible y del objetivo de cada proyecto.

Considero que estas técnicas pueden contribuir ampliamente al éxito en la implantación de sistemas y que puede extender su uso más allá de los sistemas administrativos o de negocio, ya que puede ser aplicada también a utilerías, programas-producto y software aplicativo.

Toda esta metodología tiene su razón de ser a través del beneficio que produce a organizaciones, industrias y compañías en la generación de productos y servicios de altos estándares de calidad. Las características del producto pueden ser de naturaleza diferente a las tecnológicas: la rapidez de la entrega, la facilidad de mantenimiento y la cortesía en el trato.

---

La satisfacción del cliente con el producto repercute en las ventas; las deficiencias incrementan los costos al rehacer trabajos y responder reclamaciones de los clientes. Hay productos que no se venden, a pesar de generar poca, o ninguna insatisfacción, debido a que la competencia proporciona mayor satisfacción.

En el caso de la producción de programas de computadoras, la calidad depende de factores detectados por los usuarios; por ejemplo, la velocidad o facilidad de uso, y de los factores internos sólo perceptibles por profesionales de la computación tales como el que un programa sea modular o legible. Aunque los factores externos son decisivos, los internos son los que determinan las variables y atributos de los externos.

La ingeniería de sistemas busca los factores externos de la calidad de los programas: corrección, robustez, extensibilidad, reusabilidad, compatibilidad, eficiencia, portabilidad, verificabilidad, integridad y la facilidad de uso, todo mediante la obtención de factores internos de calidad que permiten lidiar con la complejidad inherente al software industrial; es decir, la factibilidad de rehusar, reparar o extender.

Una característica del software industrial es la de ser intensamente difícil, si no imposible, para el diseñador individual comprender todas sus sutilezas. La complejidad que excede la capacidad individual para manejarla, surge de la dificultad de que los usuarios den expresión a sus necesidades de forma tal que sean comprendidos por los desarrolladores de un sistema. Igualmente surge de que durante la fase de desarrollo de un sistema, los requerimientos del mismo cambian, complicándose las labores de mantenimiento y preservación, lo que determina que la principal tarea de un equipo de desarrollo sea suministrar permanentemente, cuando menos, una ilusión de simplicidad que permita manejar la complejidad de tal sistema.

Una compañía constructora jamás enfrenta el problema de fabricar varilla, cemento y ladrillos, lo que no sucede en una compañía que produce sistemas, la cual debido a que el software, dispone de máxima flexibilidad y puede expresar cualquier abstracción, los desarrolladores de sistemas son seducidos por dicha propiedad para manufacturar desde los bloques más elementales hasta la abstracción más elevada.

Modelar sistemas continuos con un sistema discreto conduce a una complejidad adicional, ejemplificada por la siguiente situación: si se lanza una pelota al aire, es bastante fácil prever su comportamiento; sería una gran sorpresa que con un impulso adicional la pelota se detuviera a la mitad del camino y saliera disparada en dirección horizontal, sin embargo, en un software no completamente depurado, que simule el movimiento de una pelota no es nada extraño que produzca semejante resultado.

También sucede que mientras más complejo es un sistema, más expuesto queda a una falla total: un profesional de la construcción jamás dudaría del enorme costo y riesgo de cambiar la cimentación de un edificio de 50 pisos; sorprendentemente, los usuarios de software rara vez piensan dos veces en realizar cambios semejantes, pues según ellos, todo se reduce a programar.

El fracaso para dominar la complejidad del software conduce al desarrollo de proyectos demorados, fuera de presupuesto y deficientes con respecto a sus requerimientos. A lo cual tradicionalmente se le ha llamado *crisis del software*, pero un mal que dura ya décadas, más bien describe el estado normal de la industria, y es lo que sin duda, más ha contribuido al desperdicio de recursos, principalmente humanos, por estar destinados significativamente al mantenimiento y preservación geriátrico del software, con la consiguiente pérdida de oportunidades.

A lo largo de estos capítulos se ha podido plasmar una nueva técnica que esta soportada en la calidad. Calidad que busca hacer las cosas mejor en pro de la excelencia.

Esta tesis busca por sobre cualquier cosa aportar una técnica que actúe como guía en el desarrollo de aplicaciones, que casualmente es de las actividades más importantes y que marcan la pauta para nuevos desarrollos tecnológicos a nivel mundial, en el ámbito de la Industria de la computación.

Evidentemente existen ya muchas otras técnicas creadas hace ya varios años. Sin embargo, los tiempos, las necesidades, las oportunidades y el intercambio mundial son diferentes día a día; el mundo busca con mayor énfasis y con clara conciencia, una calidad en lo que obtiene y en lo que se ofrece. Y es que la calidad no esta de moda, la calidad ha estado inmersa en toda la historia desde que el mundo existe, solo que ahora se quiere entender, se desea garantizar y se intenta hacerla parte de una actitud, de una costumbre y parte de la personalidad misma de cada uno.

Estas técnicas de pruebas de sistemas persiguen el crear una calidad que satisfaga ampliamente y que por mucho, proporcione un valor agregado a quien tiene una necesidad en el proceso de información y una expectativa de cómo cubrirla.

La calidad es una energía, una estrategia de competencia para el éxito de las naciones, y en especial para nuestro México. En la economía global de hoy, la calidad esta definida en términos de los clientes y usuarios de cada producto, bien o servicio.

La creación de esta ventaja competitiva focaliza todo aquello que crea un valor agregado a todo lo que se hace, mientras que la calidad es definida, exigida y aceptada por los usuarios finales, ésta es desarrollada por ingenieros y por cada uno de los que elaboran programas de computación, porque se debe recordar que la calidad ya no la hacen los especialistas, ahora es el trabajo de todos y cada uno.

Es entendible por otro lado, que la transformación a un sistema de altos estándares de calidad total es un proceso evolutivo. Esto representa un cambio cultural hacia la persistencia, entendimiento, flexibilidad y nuevos perfiles que hagan posible su realización.

El total de estas técnicas y herramientas contenidas en esta tesis, ofrecen muchos beneficios, incluyendo la reducción de costos, mejorando las relaciones de trabajo con los usuarios y mejorando los resultados. La implantación de estas metodologías permite un mejor conocimiento de como operan las cosas, las relaciones, las actividades, conocer cómo es realmente el ambiente que nos rodea e influencia, e impulsa a hacer lo mejor de si mismo, al mismo tiempo que acrecenta los conocimientos respecto al problema que se busca resolver.

En conclusión, la clave, el punto central de los buenos resultados, es pensar y actuar en términos del usuario final; lo primero es enfocar toda la atención al cliente, y es por ello que en el capítulo 1 se describe la importancia de las pruebas en los sistemas de información. Preguntando e investigando a través de técnicas correctas, se logrará saber qué es lo importante para el usuario final y por qué quiere hacer lo que esta solicitando. Es decir, buscar la satisfacción total de quien lo requiere.

Esta satisfacción, que en términos actuales se traduce como satisfacción al cliente, no es estática, sino por el contrario es muy dinámica: lo cual significa que hoy un cliente satisfecho puede ser mañana un cliente insatisfecho y descontento.

Para evitarlo se debe ser muy cuidadoso en elaborar resultados que sean constantemente satisfactorios y que presenten mejoras continuas. Pero principalmente que cubran con las expectativas creadas por el usuario final.

## BIBLIOGRAFÍA.

- **ISO-9000.**

Brian Rothery.  
Panorama.  
Segunda Edición 1993-1996.

- **SOFTWARE TESTING.**

Bill Hetzel.  
QED Information Sciencies, Inc.  
Segunda Edición.

- **THE ECONOMICS OF SOFTWARE QUALITY ASSURANCE.**

Alberts, D.S.  
In Proceedings.  
National Computer Technology.

- **CHARACTERISTICS OF SOFTWARE QUALITY.**

Boehm, B.W.  
TRW Series of software Technology.Panorama.  
New York 1994.

- **SOFTWARE ENGINEERING ENVIRONMENTS.**

Charette, Robert N.  
McGraw Hill, Highstown.  
New York.

- **AN INTRODUCTION TO SOFTWARE QUALITY CONTROL.**

Chen, Juec Cho.  
New York, John Wiley 1992.