EVALUACION DEL IMPACTO EN TIEMPO Y RECURSOS QUE TIENE EN EL AREA DE SISTEMAS DE LAS ORGANIZACIONES LA TECNOLOGIA DE INFORMACION UTILIZADA COMO BASES DE CONOCIMIENTO, APLICADA EN LA RECUPERACION DE ERRORES DE SISTEMAS DE COMPUTO

TESIS PRESENTADA

POR

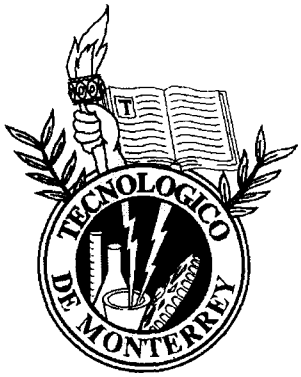
JOSE MARIA AGUILERA MENDEZ

PRESENTADA ANTE LA DIRECCION ACADEMICA DE LA UNIVERSIDAD VIRTUAL DEL INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY COMO REQUISITO PARCIAL PARA OPTAR AL TITULO DE

MAESTRO EN ADMINISTRACION DE TECNOLOGIAS DE INFORMACION

MAYO DE 1999.

EVALUACION DEL IMPACTO EN TIEMPO Y RECURSOS QUE TIENE EN EL AREA DE SISTEMAS DE LAS ORGANIZACIONES LA TECNOLOGIA DE INFORMACION UTILIZADA COMO BASES DE CONOCIMIENTO, APLICADA EN LA RECUPERACION DE ERRORES DE SISTEMAS DE COMPUTO



Tesis presentada

por

JOSE MARIA AGUILERA MENDEZ

Presentada ante la Dirección Académica de la Universidad Virtual del

Instituto Tecnológico y de Estudios Superiores de Monterrey

como requisito parcial para optar

al título de

MAESTRO EN ADMINISTRACION DE TECNOLOGIAS DE INFORMACION

Mayo de 1999

A mis padres, con respeto y cariño

¿Qué es el todo comparado con la nada?

Nada... si se compara con el todo.

"La correa de hierro de la verdad, que vosotros calificáis de invariable, os mantiene ciegos en un círculo vicioso. Técnicamente se puede tener razón en los hechos y, sin embargo, estar eternamente equivocados en la verdad"

Benítez, Juan José; Caballo de Troya 3 Planeta, México, 1990, pp. 335

RESUMEN

"EVALUACION DEL IMPACTO EN TIEMPO Y RECURSOS QUE TIENE EN EL AREA DE SISTEMAS DE LAS ORGANIZACIONES LA TECNOLOGIA DE INFORMACION UTILIZADA COMO BASES DE CONOCIMIENTO, APLICADA EN LA RECUPERACION DE ERRORES DE SISTEMAS DE COMPUTO"

MAYO, 1999.

JOSE MARIA AGUILERA MENDEZ

INGENIERO EN SISTEMAS COMPUTACIONALES

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY,

CAMPUS TOLUCA

DIRIGIDA POR LA LIC. BERTA ALICIA VALDEZ ALDRETE MCC.

El aseguramiento de la calidad del software y las pruebas para detectar errores son áreas que parecen estar en la práctica más separadas que unidas. A veces se ven como culpándose la una a la otra que por la falta de su correcta aplicación todo el proyecto fue un fracaso.

Uno de los eventos que orillan a los programadores y analistas a tratar de pasar por alto estas actividades es la constante documentación de los eventos y su recuperación en las distintas fuentes (generalmente escritas en papel) y con ello, un retraso en la puesta en marcha del proyecto; por ello decidí hacer una **evaluación del impacto en tiempo y recursos que tiene en el área de sistemas de las organizaciones la tecnología de información utilizada como bases de conocimiento, aplicada en la recuperación de errores de sistemas de computo**.

Entre los pasos para lograr la evaluación estuvieron: localizar empresas que contaran con departamentos de informática que realizarán desarrollos de sistemas o al menos su mantenimiento al nivel de código; convencer a las que no tenían un plan de seguimiento de que éste podría ser un excelente indicador de su desempeño, además de llevar desde ese momento una memoria organizacional para esa área; posteriormente, se debió obtener la información de las variables: tipo de error, tiempo de respuesta y número de incidencias; y por último aplicar el método estadístico de mínimos cuadrados para lograr un ajuste de los puntos que se presentaron en el tiempo y con ello establecer su patrón y poder determinar si crecieron o decrecieron sus tiempos.

Como quedo indicado en las tendencias (pendiente de la recta ajustada) de los tiempos de los errores, la gran mayoría, logro reducir el tiempo de respuesta a los errores y ser más efectivo en su solución. Si es verdad que un error corregido en una parte del código tiene una posibilidad mínima de volver a ocurrir, no es así el tipo de error, que puede ser reproducido "n" veces a lo largo del programa; con ello se comprueba que los recursos empleados para su manejo disminuyen de alguna manera.

INDICE

RESUMEN ..... **V**

INDICE ..... **VII**

INDICE DE TABLAS ..... **XI**

INDICE DE ILUSTRACIONES O FIGURAS ..... **XII**

CAPITULO

I. INTRODUCCION **1**

1. OBJETIVO 2

2. RESTRICCIONES 3

3. METODOLOGÍA Y MÉTODO 4

4. PRODUCTO FINAL 5

5. ORGANIZACIÓN DE LA TESIS 6

II. ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE **7**

1. ¿CÓMO SE PUEDE OBTENER UN SOFTWARE DE CALIDAD? 7

2. ¿CÓMO SE PUEDE CONTROLAR LA CALIDAD DEL SOFTWARE? 8

3. CONCEPTOS Y DEFINICIONES 9

4. ESTÁNDARES Y PROCEDIMIENTOS 9

5. ACTIVIDADES DEL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE 11

6. RELACIÓN DEL SQA CON OTRAS ACTIVIDADES DEL ASEGURAMIENTO 11

a. Monitoreo de la Administración de la configuración. 11

b. Monitoreo de validación y verificación 13

c. Monitoreo de la prueba formal 14

7. ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE DURANTE EL CICLO DE VIDA DE ADQUISICIÓN DEL SOFTWARE 15

a. Concepto de software y fase de iniciación 15
b. Fase de requerimientos de software 15
c. Fase de diseño preliminar 15
d. Fase de diseño detallado 16
e. Fase de implementación de software 16
f. Fase de prueba e integración de software 16
g. Fase de aceptación y entrega de software 17
h. Fase de operaciones e ingeniería de software 17
8. TÉCNICAS Y HERRAMIENTAS 17
9. CICLO DE VIDA DE LA CALIDAD DEL SOFTWARE 22

III. PRUEBAS EN EL SOFTWARE 26

1. DEFINICIÓN: ERROR DE SOFTWARE 26
2. DEFINICIÓN: PRUEBA DE SOFTWARE 28
3. ACTIVIDADES DE PRUEBA DURANTE EL CICLO DE DESARROLLO DE SOFTWARE 32
4. TAXONOMÍA DE LAS FALLAS 34
5. NIVELES DE PRUEBA 36
6. DOCUMENTOS DE PRUEBA 39
7. PROCEDIMIENTO PARA EL MANEJO DE ERRORES 44

IV. APRENDIZAJE ORGANIZACIONAL 47

1. ¿QUÉ ES EL APRENDIZAJE ORGANIZACIONAL? 47
2. ¿QUÉ ES UNA ORGANIZACIÓN APRENDIENTE? 48
3. ¿QUÉ ES APRENDIZAJE? 49
4. TIPOS DE APRENDIZAJE 50
5. NIVELES DE APRENDIZAJE 52
6. HOW... ENFOQUE DE LAS CINCO DISCIPLINAS PARA UNA ORGANIZACIÓN APRENDIENTE 53
a. 1ª Disciplina: Dominio Personal 54
b. 2ª Disciplina: Modelos Mentales 56
c. 3ª Disciplina: Construcción de una visión compartida 57

d. 4ª Disciplina: Aprendizaje en equipo 59
e. 5ª Disciplina: Pensamiento sistémico 59
f. Relación de las disciplinas primera y quinta 61
g. Relación de las disciplinas segunda y quinta 61
h. Relación de las disciplinas tercera y quinta 62
i. Relación de las disciplinas cuarta y quinta 62
7. MEMORIA ORGANIZACIONAL 64
a. ¿Realmente se puede almacenar el conocimiento? 65
b. La administración del conocimiento 66
c. ¿Qué es en sí la administración del conocimiento? 67
d. Tecnologías para la memoria organizacional 68
e. Una solución factible: el uso de las Tecnologías de Información 68
f. Trabajo Colaborativo Asistido por Computadora y el Groupware 71
8. RELACIÓN CON LA ADMINISTRACIÓN DEL CONOCIMIENTO 72
a. ¿Qué tecnologías de información pueden ayudarnos a llevar a cabo este almacenamiento, acceso, y reuso de la memoria organizacional? 73
b. Tecnología de Bases de Datos 73
c. Answer Garden 74
d. Case Based Reasoning 75
e. Sistemas Basados en Casos y sistemas expertos 77
f. Fuzzy Reasoning 79
g. Groupware 80
h. Intranets 90
i. Data warehouse 93
j. Aseguramiento de la calidad de los datos 98
8. POTENCIAL PARA EL APRENDIZAJE MÁS RÁPIDO 101
V. DESARROLLO DEL EXPERIMENTO 103
1. BASE DE CONOCIMIENTOS 104
2. RECOLECCIÓN Y ANÁLISIS DE LA MUESTRA 106

3. AJUSTE DE LOS DATOS A UNA RECTA 122

VI. CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS 125

1. CONCLUSIONES 125

2. RECOMENDACIONES Y TRABAJOS FUTUROS 126

ANEXOS

I. ISO 9000 128

II. SOFTWARE ASSURANCE STANDARD – NASA 133

III. NOTES ON AS 274 "LEAST-SQUARES ROUTINES TO SUPPLEMENT THOSE OF GENTLEMAN" 152

BIBLIOGRAFIA 158

INDICE DE TABLAS

TABLA 5.1. MUESTRA DE LAS DIFERENTES PLATAFORMAS DE TRABAJO. 108

TABLA 5.2. TAMAÑO DE LAS MUESTRAS. 109

TABLA 5.3. PORCENTAJES DE LOS ERRORES MÁS COMUNES CON RESPECTO A LA MUESTRA. 113

INDICE DE ILUSTRACIONES O FIGURAS

FIGURA 2.1. DIAGRAMA DE FLUJO DE UNA AUDITORÍA BÁSICA. 21

FIGURA 2.2. EL CICLO DE VIDA DE LA CALIDAD DEL SOFTWARE. 23

FIGURA 3.1. EJEMPLO DE PROGRAMA TRIVIAL. 28

FIGURA 3.2. MODELO "V" DEL CICLO DE DESARROLLO DEL SOFTWARE. 33

FIGURA 3.3. COSTO DE DETECTAR Y CORREGIR PROBLEMAS DE SOFTWARE. SIGNIFICADO DE LAS ABREVIACIONES: PM – PRUEBA DE MÓDULO, PI – PRUEBA DE INTEGRACIÓN, PS – PRUEBA DEL SISTEMA, PA – PRUEBA DE ACEPTACIÓN. 35

FIGURA 3.4. DOCUMENTOS DE PRUEBA EN EL MODELO "V" DEL CICLO DE DESARROLLO DE SOFTWARE. 39

FIGURA 3.5. RELACIÓN DE LOS DOCUMENTOS DE PRUEBA CON EL PROCESO DE PRUEBA. 41

FIGURA 3.6. PRUEBA DE DOCUMENTOS UTILIZANDO LA ESTRUCTURA INTERNA DE TEAMWARE. 44

FIGURA 3.7. MANEJO DE ERRORES UTILIZANDO UN SISTEMA DE SEGUIMIENTO. 45

FIGURA 4.1. CÍRCULO DEL APRENDIZAJE. 50

FIGURA 4.2. NIVELES DE LA PRIMERA DISCIPLINA. DOMINIO PERSONAL. 55

FIGURA 4.3. NIVELES DE LA SEGUNDA DISCIPLINA. MODELOS MENTALES. 56

FIGURA 4.4. NIVELES DE LA TERCERA DISCIPLINA. CONSTRUCCIÓN DE LA VISIÓN COMPARTIDA. 57

FIGURA 4.5. EJEMPLO BÁSICO DE UN ARQUETIPO SISTÉMICO, "LÍMITES DEL CRECIMIENTO" 58

FIGURA 4.6. NIVELES DE LA CUARTA DISCIPLINA. APRENDIZAJE EN EQUIPO. 60

FIGURA 4.7. NIVELES DE LA QUINTA DISCIPLINA. PENSAMIENTO SISTÉMICO. 60

FIGURA 4.8. MAPA DE INFLUENCIAS O ARQUETIPO DE UNA ORGANIZACIÓN APRENDIENTE. ESTE MAPA SOLO MUESTRA ALGUNAS DE LAS VARIABLES INTERPERSONALES INVOLUCRADAS, TOMANDO COMO PUNTO DE PARTIDA EL PAPEL DE LOS LÍDERES, COMO PROMOTORES DE UNA CULTURA BASADA EN EL DIÁLOGO, EL CUESTIONAMIENTO Y EL TRABAJO EN EQUIPO. 64

FIGURA 4.9. DIAGRAMA EVOLUTIVO. 72

FIGURA 4.10. ANSWER GARDEN DE ACKERMAN. 74

FIGURA 4.11. EL CICLO DE RBC. 75

FIGURA 4.12. PROCESO DE EXTRACCIÓN DEL CONOCIMIENTO. 78

FIGURA 4.13. SISTEMA PARA LA ADMINISTRACIÓN DE INFORMACIÓN DISTRIBUIDA (DIMS), FUENTE: PLUMTREE SOFTWARE. 85

FIGURA 4.14. EL MEDIO AMBIENTE DEL GROUPWARE. 86

FIGURA 4.15. PLANES CORPORATIVOS PARA EL USO DE INTRANET, FUENTE: FORRESTER RESEARCH. 1996. 92

FIGURA 4.16. ARQUITECTURA DEL DATA WAREHOUSE 96

FIGURA 4.17.EL PROCESO DE CONVERSIÓN DE DATOS. © 1997 MILLER FREEMAN, INC. 99

GRÁFICO 5.1. DISTRIBUCIÓN DE ERRORES EN EL SISTEMA DE NÓMINA 110

GRÁFICO 5.2. DISTRIBUCIÓN DE ERRORES DEL SISTEMA DE CUENTAS POR PAGAR. 110

GRÁFICO 5.3. DISTRIBUCIÓN DE ERRORES EN EL SISTEMA DE MATERIALES (MATISA). 111

GRÁFICO 5.4. DISTRIBUCIÓN DE ERRORES DEL SISTEMA DE CONTABILIDAD 111

GRÁFICO 5.5. DISTRIBUCIÓN DE ERRORES DEL SISTEMA DE MATERIALES (IUSA) 112

GRÁFICO 5.6. DISTRIBUCIÓN DE ERRORES SISTEMA DE VENTAS. 112

GRÁFICO 5.7. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE NÓMINA. 116

GRÁFICO 5.8. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE NÓMINA Y SUS INCIDENCIAS. 116

GRÁFICO 5.9. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE CUENTAS POR PAGAR. ... 117

GRÁFICO 5.10. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE CUENTAS POR PAGAR Y SUS INCIDENCIAS. 117

GRÁFICO 5.11. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE MATERIALES (MATISA). 118

GRÁFICO 5.12. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE MATERIALES Y SUS INCIDENCIAS (MATISA). 118

GRÁFICO 5.13. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE CONTABILIDAD. 119

GRÁFICO 5.14. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE CONTABILIDAD Y SUS INCIDENCIAS. 119

GRÁFICO 5.15. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE MATERIALES (IUSA)... 120

GRÁFICO 5.16. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE MATERIALES Y SUS INCIDENCIAS (IUSA). 120

GRÁFICO 5.17. COMPORTAMIENTO DE LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE VENTAS. **121**

GRÁFICO 5.18. FECHAS EN QUE OCURRIERON LOS TRES PRINCIPALES ERRORES DEL SISTEMA DE VENTAS Y SUS INCIDENCIAS. 121

CAPITULO I

INTRODUCCION

Comparto la visión con otros autores de que el aprendizaje organizacional se presenta como una evolución natural de las políticas japonesas del aseguramiento de calidad que nacieron a inicios de los años 80's y que siguen teniendo, hasta la fecha, una validez extraordinaria como herramientas que se utilizan para lograr que una empresa sea totalmente aprendiente, o al menos y a gusto de la administración, se sientan las bases para lograrlo; aunque sería tonto no continuar con el esfuerzo o dejarlo a medias después de todo el trabajo realizado.

Ahora definamos que es una **organización aprendiente:** organización con habilidades de crear, adquirir, transferir y modificar su conducta para reflejar su conocimiento e incluir y aplicar las nuevas ideas en la empresa.

Este proceso como se puede pensar no es nada sencillo ya que implica, muchas veces, en principio un cambio de actitud en las personas que integran la organización y mucho trabajo por parte de todos debido a la documentación que se tiene que realizar de los procesos; por su parte la alta administración no puede esperar que estos cambios sean fáciles y mucho menos ver los resultados en el corto plazo.

El aprendizaje organizacional es un proceso que tiene mucho parecido al aprendizaje humano, con la única diferencia de que el hombre deja mucho de su conocimiento a un nivel intangible, es decir, se queda en su mente y nunca es registrado en un medio físico como papel, cinta, disco magnético, etc. en cambio la condición para que se considere que una organización posea memoria es el hecho de registrar todos los acontecimientos, procesos, dificultades y demás con sus respectivas observaciones y soluciones pertinentes; esto como es de esperarse hará que en

un futuro el valor de las empresas también se mida por su memoria organizacional y no sólo por el valor de sus activos y productos.

Un problema que se puede solucionar utilizando el enfoque de las organizaciones aprendientes es el desarrollo de software; éste comprende a grandes rasgos: análisis, diagrama de flujo, lenguaje estructurado, programación, pruebas y mantenimiento; ahora aplicando las herramientas de la tecnología de información se pueden crear bases de conocimientos que ayuden a los analistas, programadores y gente de soporte técnico a resolver los problemas más rápido y darles seguimiento de forma ágil y objetiva.

Se entiende como bases de conocimiento a un programa computacional que registra un problema o situación y que de acuerdo a como el usuario le introduzca información sobre el seguimiento del problema, propuestas, observaciones y posibles soluciones; ésta se va "colgando" o relacionando al tópico inicial. La organización de éstas bases de conocimiento lo que buscan es agilizar la solución de un problema específico y al mismo tiempo conservarlo y compartirlo con los demás.

Un estudio para determinar el impacto que tiene el uso de éste tipo de herramientas en las organizaciones puede dar evidencias más concretas sobre su utilidad y los beneficios que nos darían su manejo adecuado, además que nos servirá para determinar si estas herramientas sobrevivirán o no al próximo siglo.

1. Objetivo

Como ya se mencionó anteriormente, el impacto que esta teniendo la aplicación de tecnologías de información a las organizaciones aprendientes se puede considerar como muy positivo y esta llevando a muchas empresas a elevar su productividad.

Sin embargo su uso en la industria del software aún no es muy conocido o no ha sido lo suficientemente explotado, por lo que el objetivo del siguiente trabajo será:

"Evaluar el impacto en tiempo y recursos que tiene en el área de sistemas de las organizaciones la Tecnología de Información utilizada como bases de conocimiento, aplicada en la recuperación de errores de sistemas de cómputo."

Entendiendo como recursos a:

1. Horas programador
2. Horas analista
3. Horas computadora (máquina)
4. Tiempos muertos por caídas del sistema

2. Restricciones

Se definen las siguientes restricciones:

- La investigación se realizó en empresas localizadas en la periferia de la ciudad de Toluca, estado de México y en el ITESM Campus Toluca.
- Sólo aplicaron aquellas empresas que tenían un departamento de sistemas que cuente con un área de desarrollo y que trabajen bajo esquemas de organizaciones aprendientes o estén certificados en el área, con cualquier estándar de calidad (ISO, etc.).
- Se evaluó el software que fue susceptible de mantenimiento, es decir, que tuvo la facilidad de modificar los códigos fuente y ponerlos en operación después de corregir la falla.
- Debido a la falta de material bibliográfico con referencia al área de auditoria y control de sistemas de información, la mayor parte de la investigación se basó en los libros: EDP Auditing de Ron Weber y Auditing EDP Systems de Javier Kuong y Software System Testing and Quality Assurance de Boris Beizer.

- La investigación de campo se delimito al tiempo que transcurrió entre los meses de enero y octubre de 1998, que se refiere a investigación de campo marcado en el plan de estudios y avances.
- Los errores de sistema que se registraron fueron los generados a momento de ejecución o los detectados por los programadores y/o analistas en código fuente.
- La recolección de datos en las empresas fue cada quince días; comenzando en la primera quincena de enero de 1998.

3. Metodología y método

Para realizar la investigación se requiere de una metodología que cuente con características enfocadas al investigador tales como: enfocarse en hechos, formular hipótesis y probarlas, ver causas y leyes fundamentales, reducir los problemas a algo simple. Desde el punto de vista de los métodos que se utilizarán debe cumplir con la operacionalización de los conceptos para que éstos puedan ser medidos.

De acuerdo al objetivo del estudio, que se basa en la evaluación de un fenómeno para determinar si es benéfica o no su aplicación y las características que se requieren de él para realizar la investigación, la selección fue el método **cuantitativo**; este método se basa en la premisa de que el universo existe y sus atributos deben ser medidos utilizando métodos objetivos; además es ampliamente utilizado para responder a preguntas del tipo: qué, cuáles y cuántos. La utilización de este método nos da algunas ventajas, como es el hecho de que es económico y rápido de aplicar ya que utiliza datos cuantitativos que son fáciles de recolectar y analizar; otra ventaja es que los estudios sobre muestras grandes son muy importantes para la obtención de políticas generales. Por otra parte, tiene algunas desventajas entre las que destacan el hecho de que los métodos estadísticos son en muchas ocasiones inflexibles y a veces artificiales; además de que no son métodos efectivos en el entendimiento del significado que las personas dan a sus acciones.

Dentro de la metodología cuantitativa existen 4 métodos para la obtención de resultados. Todas buscan la manipulación de variables independientes (variables cuyo efecto en otras variables se puede medir, el investigador tiene dominio sobre ellas) e intervinientes (variables que afectan el comportamiento de las variables dependientes, pero cuyo efecto no se desea medir) para medir su impacto en las variables dependientes (variables cuyo comportamiento se analiza, considerando la incidencia del efecto de las variables independientes, el investigador no tiene control sobre ellas), a través del grupo experimental (grupo que se estudiará, haciéndolo pasar por las incidencias de las variables independientes). En esta investigación se utilizará el método de **investigación de acción** por sus características de libertad por parte del investigador de intervenir en el grupo experimental definido y por proporcionar resultados más concretos a una situación, como la es el medir el impacto en tiempo y recursos que tendría la utilización de bases de conocimiento en la recuperación de errores de software.

4. Producto final

Al concluir la investigación se obtendrán dos indicadores considerados vitales en las áreas de mantenimiento de sistemas y soporte técnico: **tiempo y recursos empleados**. Estos indicadores en conjunto son una excelente fuente para evaluar el desempeño de las áreas; y por lo general tienden a indicar quién hace uso más eficiente de los elementos con que se cuenta; claro que éste no es el objetivo de la investigación, lo que se busca desde un principio es dar un panorama general del posible beneficio del uso de herramientas de tecnologías de información y cómo su correcto mantenimiento puede ayudarnos a hacer más productivo nuestro trabajo.

Con la elaboración de esta tesis espero contribuir de tal manera que sirva como un formador o reforzador de las políticas de uso y mantenimiento de bases de conocimiento en los centros de soporte, análisis y desarrollo de sistemas de las organizaciones que cuenten con la infraestructura necesaria para poder explotarlas o, en su defecto, motivar a aquellas que no

cuenten con los recursos informáticos necesarios a habilitar los mecanismos básicos (en papel o base de datos) para llevar una memoria departamental.

5. Organización de la tesis

La tesis consta de 6 capítulos organizados de la siguiente manera:

- **Capitulo I. Introducción.**
- **Capítulo II. Aseguramiento de la calidad del software**, en donde se da una breve explicación de lo que es la calidad de software y algunas herramientas que facilitan su obtención.
- **Capítulo III. Pruebas en el software**, en este capítulo tratamos de definir: ¿qué es un error? y ¿cómo evitar que nuestro software tenga errores?
- **Capítulo IV. Aprendizaje organizacional**, esta es una rápida explicación de lo que se entiende por empresa u organización aprendiente y las herramientas que existen en el mercado para facilitar su administración.
- **Capítulo V. Desarrollo del experimento.**
- **Capitulo VI. Conclusiones, recomendaciones y trabajos futuros.**

Los diferentes temas se relacionan debido a que para desarrollar un software más barato, según Beizer y Myers, se deben de minimizar los gastos generados por los errores que tenga el software; entonces debemos generar un software con calidad que contenga todo su proceso de elaboración bien entendido y documentado para, con esas bases, lograr pruebas más estrictas que nos aseguren un software 100% libre de errores. Ahora bien, todo ese conocimiento debe ser más abierto, o en otras palabras, todos los que participan de la elaboración de un producto deben tener acceso a fuentes de información que indiquen los errores, trucos, trampas, defectos, etcétera; que pueden existir o que han ocurrido en todo el proceso de fabricación de software; con ello el tiempo de respuesta es menor y la efectividad para resolver un problema es mayor; logrando con ello un impacto positivo (disminución de los gastos) en los recursos de las empresas.

CAPITULO II

ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

Uno de los problemas más serios que enfrentan las compañías que desarrollan software es la calidad de sus productos. Desde la década de los 70's, este tema ha sido motivo de preocupación para especialistas, investigadores y distribuidores; los cuales han realizado una gran cantidad de pesquisas al respecto con dos objetivos fundamentales:

1. ¿Cómo obtener un software con calidad?
2. ¿Cómo evaluar la calidad del software?

Las dos preguntas llevan implícitas varias y amplias respuestas, pero están estrechamente relacionadas con el concepto de la calidad del software, que es resultado de la primera y fuente de la segunda.

1. ¿Cómo se puede obtener un software de calidad?

La obtención de un software con calidad implica la utilización de métodos o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, para lograr con ello una mayor confiabilidad, manutención y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La política establecida debe estar sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.

- El **principio tecnológico** define las técnicas a utilizar en el proceso de desarrollo del software.

- El **principio administrativo** contempla las funciones de planificación y control del desarrollo del software, así como la organización del ambiente o centro de ingeniería de software.
- El **principio ergonómico** define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, **pero no la asegura**. Para el aseguramiento de la calidad es necesario su control o evaluación.

2. ¿Cómo se puede controlar la calidad del software?

Para controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Beizer, "usted no puede controlar lo que no se puede medir"[1].

Las cualidades para medir la calidad del software son definidas por innumerables autores, los cuales las denominan y agrupan de formas diferentes. Por ejemplo, Myers define métricas de calidad y criterios, donde cada métrica se obtiene a partir de combinaciones de los diferentes criterios. Otros autores identifican la calidad con el nivel de complejidad del software y definen dos categorías de métricas: de complejidad de programa o código, y de complejidad de sistema o estructura.

Todos los autores coinciden en que el software posee determinados índices que pueden ser medidos y que son las bases para la calidad, el control y el perfeccionamiento de la productividad.

---

[1] Beizer, Boris. Software System Testing and Quality Assurance, Van Nostrand Reinhold Co., New York, 1984, pp. 17.

Una vez seleccionados los índices de calidad, se debe establecer el proceso de control, que requiere los siguientes pasos:

- **Definir el software** que va a ser controlado: clasificación por tipo, esfera de aplicación, complejidad, etc., de acuerdo con los estándares establecidos para el desarrollo del software.
- **Seleccionar una medida** que pueda ser aplicada al objeto de control. Para cada clase de software es necesario definir los indicadores y sus magnitudes.
- **Crear o determinar los métodos de valoración** de los indicadores, que son métodos manuales (como cuestionarios o encuestas estándares) para la medición de criterios periciales y herramientas automatizadas para medir los criterios de cálculo.
- **Definir las regulaciones** de la empresa para realizar el control: quiénes participan en el control de la calidad, cuándo se realiza, qué documentos deben ser revisados y elaborados, etc.

3. Conceptos y definiciones

El aseguramiento de la calidad del software (SQA) se define como un enfoque planeado y sistemático para evaluar la calidad y el cumplimiento de los estándares, procesos y procedimientos establecidos para un producto de software. El SQA incluye los procesos para asegurar que los estándares y procedimientos han sido establecidos y se siguen a través del ciclo de vida de adquisición del software, el cumplimiento de los mismos, se evalúa a través del monitoreo de procesos, la evaluación del producto y las auditorias. El desarrollo del software y el control de procesos deben incluir puntos o temas aprobados para el aseguramiento de la calidad, donde una evaluación de SQA del producto puede ser realizada, tomando en cuenta, los estándares aplicables.

4. Estándares y procedimientos

El establecer los estándares y procedimientos para el desarrollo del software es una de las partes más críticas, ya que éstos definen el marco de referencia para su análisis y codificación. Los

estándares son los criterios marcados contra los que se podrán comparar los productos de software, a su vez los procedimientos son los criterios establecidos contra los que se podrán comparar los procesos de desarrollo y control. Los estándares y procedimientos rigen los métodos prescritos para el desarrollo del software; el papel del SQA es asegurar su existencia y correcta adecuación. Se necesita una documentación apropiada de los estándares y procedimientos por la razón de que algunas de las actividades de SQA (monitoreo de procesos, evaluación del producto y auditoría) dependen de definiciones concretas para medir el cumplimiento del proyecto. Los tipos de estándares incluyen:

- **Estándares de documentación**: especifican la forma y contenido para la documentación de la planeación, el control y el producto y son los que mantienen la consistencia durante el proyecto.
- Los **estándares de diseño**: definen la forma y el contenido del producto diseñado; éstos dan las reglas y métodos para trasladar los requerimientos del software al diseño y para representarlos en la documentación del diseño.
- Los **estándares del código**: determinan el lenguaje con el cual será escrito el código y definen cualquier restricción en el uso de atributos del lenguaje. Además especifican las estructuras del lenguaje, convenciones de estilos, reglas, tanto para estructuras de datos como interfaces y documentación interna o comentarios en el código.

Los procedimientos son una serie de pasos que de ser seguidos pueden llevar a cabo un proceso. Todos los procesos deben tener sus procedimientos documentados. Algunos ejemplos de procesos que necesitan procedimientos son los siguientes: reporte de fallos, reportes de no conformidades, correcciones, inspecciones formales, pruebas, administración de la configuración, etc.

5. Actividades del aseguramiento de la calidad del software

La evaluación del producto y el monitoreo de los procesos son las actividades del SQA que verifican que los procesos de desarrollo y control de software (descritos en el plan maestro del proyecto), sean llevados a cabo de manera correcta y que además se apliquen los procedimientos y estándares del proyecto. Los productos son monitoreados para verificar que se cumplan los estándares y los procesos para verificar que se cumplan los procedimientos.

La evaluación del producto es una actividad del SQA que asegura que se han seguido los estándares indicados. De manera ideal, los primeros productos monitoreados por el SQA deben ser los procedimientos y estándares del proyecto. El SQA checa que existan estándares claros y realizables, posteriormente evalúa que éstos se cumplan en el producto desarrollado. La evaluación del producto verifica que el software refleje los requerimientos de los estándares aplicables como están descritos en el plan maestro.

El monitoreo del proceso es una actividad del SQA que asegura que se hayan aplicado los pasos correctos para ejecutar un proceso. El SQA monitorea los procesos al comparar los pasos ejecutados hasta el momento contra los que se deben hacer ó están documentados en los procedimientos. La sección de aseguramiento del plan maestro especifica los métodos a ser usados por la actividad de monitoreo del proceso de SQA.

6. Relación del SQA con otras actividades del aseguramiento

Algunas de las relaciones más importantes del SQA con otras actividades de aseguramiento y administrativas son:

a. Monitoreo de la Administración de la configuración.

El SQA asegura que las actividades de la administración de la configuración (CM, por sus siglas en inglés) se lleven a cabo de acuerdo al plan, estándares y procedimientos del CM. El SQA revisa los planes de CM para cumplir con las políticas y requerimientos del CM del software y dar un

seguimiento para las no conformidades. El SQA audita las funciones del CM para checar cuánto se apegan a los estándares y procedimientos y prepara los reportes de sus observaciones.

Las actividades de CM monitoreadas y auditadas por SQA se integran por el control base, la identificación de la configuración, el control de la configuración, el registro del status de la configuración, y la confirmación de la configuración; además el SQA monitorea y audita las librerías de software.

El SQA se asegura de que:

- **Las bases han sido establecidas** y mantenidas de forma consistente, para utilizarlas en desarrollos y controles subsecuentes.
- **La identificación de la configuración** del software es consistente y adecuada con respecto a la numeración o nombre de los programas de cómputo, módulos de software, unidades de software y documentos de software asociados.
- **El control de la configuración** es mantenido como el software de configuración (utilizado en las fases críticas de prueba, aceptación y entrega); siendo compatible con la documentación asociada.
- **El registro del status de configuración** es llevado de forma adecuada incluyendo el registro y reporte de datos que reflejen la identificación de la configuración del software, cambios propuestos a la identificación de la configuración y el status de la implementación de cambios aprobados.
- **La comprobación de la configuración** del software es establecida por una serie de revisiones y auditorias a la configuración que indican el desempeño que se necesita por las especificaciones y configuración del software que esta debidamente reflejada en los documentos de diseño de software.
- **Las librerías de desarrollo** de software dan un mejor manejo del código del software, de la documentación, de los medios e información relacionada en sus variadas formas y

versiones, desde el tiempo de su aprobación inicial o aceptación, hasta su incorporación a la distribución final.

- Los cambios aprobados al software base se realicen de manera apropiada y consistente en todos los productos y que no sean realizados cambios no autorizados.

b. Monitoreo de validación y verificación

EL SQA asegura las actividades de validación y verificación (V y V) al monitorear las revisiones, inspecciones y pequeñas pruebas. El papel del SQA en las pruebas formales es descrito en la sección siguiente. El papel del SQA en las revisiones, inspecciones y pruebas es el de observar, participar según se necesite y verificar que hayan sido conducidos y documentados adecuadamente. El SQA también asegura que cualquier acción requerida sea asignada, documentada, calendarizada y actualizada.

Las revisiones formales de software deben ser enviadas al final de cada fase del ciclo de vida para identificar los problemas y determinar cuándo el producto en desarrollo cumple con todos los requerimientos necesarios. Algunos ejemplos de revisiones formales son las revisiones de diseño preliminar (PDR), revisión de diseño crítico (CDR) y revisión de pruebas de disponibilidad (TRR). Una revisión ve una imagen de todo el producto a ser desarrollado para verificar si satisface sus requerimientos.

Las revisiones son la parte del proceso de desarrollo diseñados para dar una decisión de listo/no listo para continuar con la siguiente fase. En revisiones formales, el trabajo actual es comparado con los estándares establecidos. El objetivo principal del SQA en las revisiones es asegurar que se ha seguido el plan maestro y de desarrollo y que el producto está listo para pasar a la siguiente fase de desarrollo. Aunque la decisión de continuar es una decisión administrativa, el SQA es responsable de recomendar a la administración y de participar en la decisión.

Una inspección o prueba es una revisión detallada de un producto basado en una filosofía de paso a paso, línea por línea, para localizar errores. En las inspecciones, el SQA asegura como mínimo, que el proceso ha sido completado y que se le ha dado seguimiento. El proceso de inspección puede ser utilizado para medir el cumplimiento con los estándares.

c. Monitoreo de la prueba formal

El SQA asegura que la prueba formal de software, como la prueba de aceptación, es hecha de acuerdo a los planes y procedimientos. El SQA verifica que la documentación de prueba tenga congruencia con los estándares. La verificación de los documentos incluye planes, especificaciones, procedimientos y reportes de las pruebas. El SQA monitorea las pruebas y da seguimiento a las no conformidades. El monitoreo de pruebas en el SQA asegura que el software está completo y listo para su entrega.

Algunos de los objetivos del SQA al realizar un monitoreo de las pruebas del software, es el asegurar que:

- Los procedimientos de prueba verifiquen los requerimientos del software de acuerdo con los planes de prueba.
- Los procedimientos de prueba puedan ser verificados.
- La versión definitiva del software esta siendo probada (por el monitoreo de SQA de la actividad CM).
- Se han seguido los procedimientos de prueba.
- Los errores y no conformidades ocurridas durante las pruebas (esto es, cualquier incidente no esperado en los procedimientos de prueba) han sido documentadas y registrados.
- Los reportes de prueba son correctos y están completos.
- La prueba de regresión ha sido conducida para asegurarse de que los errores y no conformidades han sido corregidas.
- La resolución de todos los errores y no conformidades se han hecho antes de la liberación.

Las pruebas de Software verifican que el software cumpla con sus requerimientos. La calidad de la prueba es el asegurar que la verificación de los requerimientos del proyecto han sido satisfechos y que el proceso de pruebas esta en concordancia con los planes y procedimientos de prueba.

7. Aseguramiento de la calidad del software durante el ciclo de vida de adquisición del software

Como refuerzo a las actividades generales descritas en las secciones E y F, existen más actividades específicas del SQA que deben ser aplicadas a ciertas fases del ciclo de vida de adquisición del software. Al final de cada fase, la aceptación por parte de SQA es un elemento clave en la decisión administrativa para iniciar la siguiente fase del ciclo de vida. Las actividades sugeridas para cada fase son las siguientes:

a. Concepto de software y fase de iniciación

El SQA debe ser integrado tanto en la escritura como en la revisión del plan maestro para asegurar que los procesos, procedimientos y estándares identificados en el plan son apropiados, claros, específicos y auditables. Durante esta fase, el SQA también da la sección de QA del plan maestro.

b. Fase de requerimientos de software

Durante la fase de requerimientos del software, el SQA asegura que los requerimientos de software están completos, probados y expresados de acuerdo a los requisitos de funcionalidad, desempeño e interfaces.

c. Fase de diseño preliminar

Las actividades del SQA durante la fase preliminar del diseño del software son:

- Asegurar el cumplimiento de los estándares de diseño aprobados en el plan maestro.
- Asegurar que todos los requerimientos del software se encuentran aplicados en los componentes del software.
- Asegurar que la matriz de verificación de pruebas existe y está actualizada.
- Asegurar que los documentos de control de interfaces cumplen con los estándares de forma y contenido.
- Revisar la documentación de PDR y asegurar que todas las acciones han sido resueltas.
- Asegurar que el diseño aprobado está colocado en la administración de la configuración.

d. Fase de diseño detallado

Las actividades de SQA durante la fase de diseño detallado incluyen:

- Asegurar que se siguen los estándares de diseño aprobados.
- Asegurar que los resultados de las inspecciones al diseño sean incluidos en el diseño.
- Revisar la documentación de CDR y asegurarse que todas las acciones han sido resueltas.

e. Fase de implementación de software

Las actividades de SQA durante la fase de implementación incluyen la auditoría de:

- Resultados de las actividades de codificación y diseño incluyendo el programa descrito en el plan de desarrollo de software.
- El status de todos los elementos a entregar.
- Las actividades de la administración de la configuración y las librerías de desarrollo de software.
- Reportes de error y no conformidades y sistema de acciones correctivas.

f. Fase de prueba e integración de software

Las actividades de SQA durante la fase de prueba e integración incluyen:

- Asegurar la disponibilidad para la prueba de todos los elementos.

- Asegurar que todas las pruebas son ejecutadas de acuerdo a los planes y procedimientos de prueba y que cualquier inconformidad, ha sido reportada y resuelta.
- Asegurarse que los reportes de prueba están completos y correctos.
- Certificar que las pruebas han terminado y que el software y la documentación están listos para entregarse.
- Participar en la revisión de la disponibilidad de las pruebas y asegurar que todas las acciones han sido terminadas.

g. Fase de aceptación y entrega de software

Las actividades de SQA durante la fase de aceptación y entrega de software son como mínimo, el asegurar la aprobación de la auditoria de configuración final, para así demostrar que todos los elementos del paquete están listos para ser entregados.

h. Fase de operaciones e ingeniería de software

En esta fase, podrían incluirse ciclos de mini-desarrollos para corregir o mejorar el software; durante estos ciclos de desarrollo, el SQA conduce las actividades especificas de las fases descritas anteriormente.

8. Técnicas y herramientas

El SQA debe evaluar sus necesidades de herramientas de aseguramiento contra aquellas disponibles en el mercado que se puedan adaptar al proyecto, desarrollando otras, en caso de ser necesario.

La mayoría de los autores de SQA concuerdan en que la mejor herramienta para lograr la calidad de un producto es la prueba del mismo, hacerlo que funcione bajo circunstancias críticas y no tan críticas para emitir los reportes de fallos (si los hay). Se debe tener la suficiente cantidad de

equipos y personal de prueba, documentación en línea, así como equipos para localizar las fallas y también suficientes programadores y diseñadores.

Una técnica fundamental del SQA es la auditoria (ver figura 2.1), que ve a profundidad dentro de un proceso y/o producto, comparándolos contra los procedimientos y estándares establecidos. La auditoria es la técnica clave utilizada para evaluar al producto y monitorear el proceso; éstas se usan para revisar los procesos administrativos, técnicos y de aseguramiento dando un indicador de la calidad y de la situación del producto de software. Las revisiones del plan maestro deben asegurar que los puntos de aprobación aplicables del SQA estén integrados a estos procesos.

El propósito de una auditoria de SQA es asegurarse de que los procedimientos de control han sido aplicados correctamente, que la documentación ha sido actualizada y que los reportes de status del programador, reflejan la situación de las actividades adecuadamente. El producto de la SQA es un reporte de auditoria para la administración que consiste en recomendaciones y observaciones hechas para llevar el desarrollo en conformidad con los estándares y /o procedimientos.

La auditoria como método para lograr la calidad se auxilia de un gran número de herramientas de soporte, como son:

- **Referencias cruzadas.** No existe un número determinado de referencias, pero un buen generador debe permitir establecer cualquier cantidad de relaciones además de admitir varios ordenamientos.
- **Revisiones de estilo y estándares.** Permite la definición de reglas de estilo, estándares y convenciones de nombre y codificación. Esta herramienta checa directamente el código para ver su cumplimiento.

- **Generadores de grafos y procesos.** Generadores que, para un análisis estático del producto generan diagramas de flujo a partir de los códigos y mejor aún, en un nivel de abstracción generan grafos de los procesos.
- **Tests drivers.** Los tests drivers son indispensables cuando no se tiene un plan de trabajo de la prueba. Los drivers organizan los casos de prueba, inicializan el sistema o la rutina para conocer sus condiciones, sus entradas y dan predicciones de los resultados basándose en los que se han generado.
- **Herramientas de instrumentación.** Son porciones de código de control que son insertadas en el sistema y que arrojan información sobre el estado de los procesos, datos, etc.

Aunque Deutsch menciona tres maneras sencillas de lograr la calidad del software:

1. **La prueba contra los errores** es el más viejo de los tres métodos y por ende el que más conocen los programadores. Probar el software significa, ejecutar el software bajo situaciones simuladas y ver si falla o no (ver capítulo 2). Como la prueba depende de la ejecución de un programa, ésta ocurre después de que el software ha sido especificado, diseñado y codificado. Aunque la prueba es una fase crítica en el logro de un software con calidad, ésta no es suficiente: "software that is not maintainable will not improve merely as a result of more testing"[2].
2. **Revisar por defectos** es un método que se hizo popular en los años 70's. Revisar el software significa someter los diseños y código fuente a inspecciones manuales para verificar que se cumple con los estándares y especificaciones. Una no-conformidad encontrada durante una inspección es un defecto. Las revisiones son conducidas por el diseño una vez que éste se ha terminado; y por el código después que ha terminado la programación. De nuevo, aunque la revisión es una fase crítica para el logro de un

[2] Deutsch, Michael; Willis, Ronald. Software Quality Engineering, Prentice-Hall, New Jersey, 1998, pp. 15.

software con calidad, ésta no es suficiente: algunos niveles de seguridad no se pueden lograr por muchas revisiones que sean hechas.

3. La más reciente adición de los enfoques para lograr un software con calidad es la **ingeniería de calidad**. Aplicar ingeniería en la calidad significa agregar calidad al software durante el proceso de ingeniería. Para lograr esto, los ingenieros y programadores deben estar consientes de los requerimientos de calidad al mismo tiempo en que trabajan en los requerimientos funcionales, entonces los requerimientos de calidad tienen el mismo peso específico en el producto como los requerimientos funcionales. Es en la ingeniería en calidad donde se debe poner un mayor énfasis. Si ésta se hace bien, esto es, si agregamos calidad al producto al momento de construirlo, entonces la necesidad de revisar y probar se reduce. La ingeniería de la calidad se enfoca en la necesidad de producir calidad durante el mismo proceso de diseño.

Aunque la ingeniería en la calidad es fundamental, ningún programa de calidad puede ser completo sin la revisión y la prueba (los humanos nunca seremos perfectos). Existen niveles de calidad que sólo se pueden lograr a través de la inspección o de la ejecución del software en su ambiente computacional.

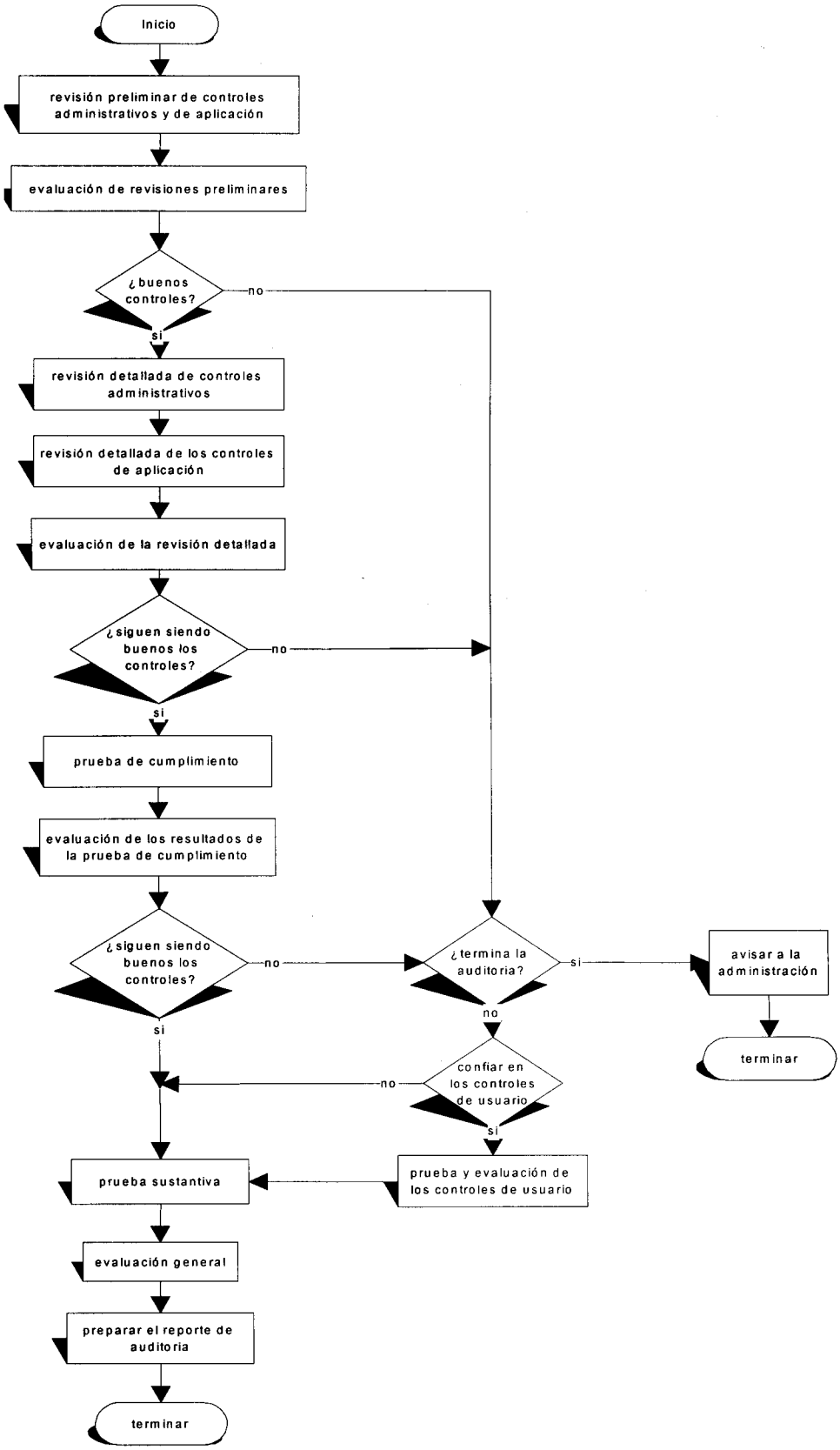


Figura 2.1. Diagrama de flujo de una auditoría básica.

9. Ciclo de vida de la calidad del software

La relación de las tres maneras de lograr la calidad descritas anteriormente se muestra en la figura 2.2. Este refleja el punto de vista de una persona relacionada con la calidad, de lo que es el ciclo de vida del desarrollo de software. La primera fase del ciclo es aplicar técnicas de ingeniería para encontrar una solución. Esta actividad transforma los requerimientos funcionales y de calidad del software en diseños y código; y es supervisada por el presupuesto y el calendario; las técnicas y herramientas de ingeniería de software son utilizadas como mecanismos para lograrlas. Al utilizar una solución de ingeniería buscamos permear la calidad utilizando técnicas y herramientas que resulten en un código y diseño que tengan los atributos requeridos de calidad.

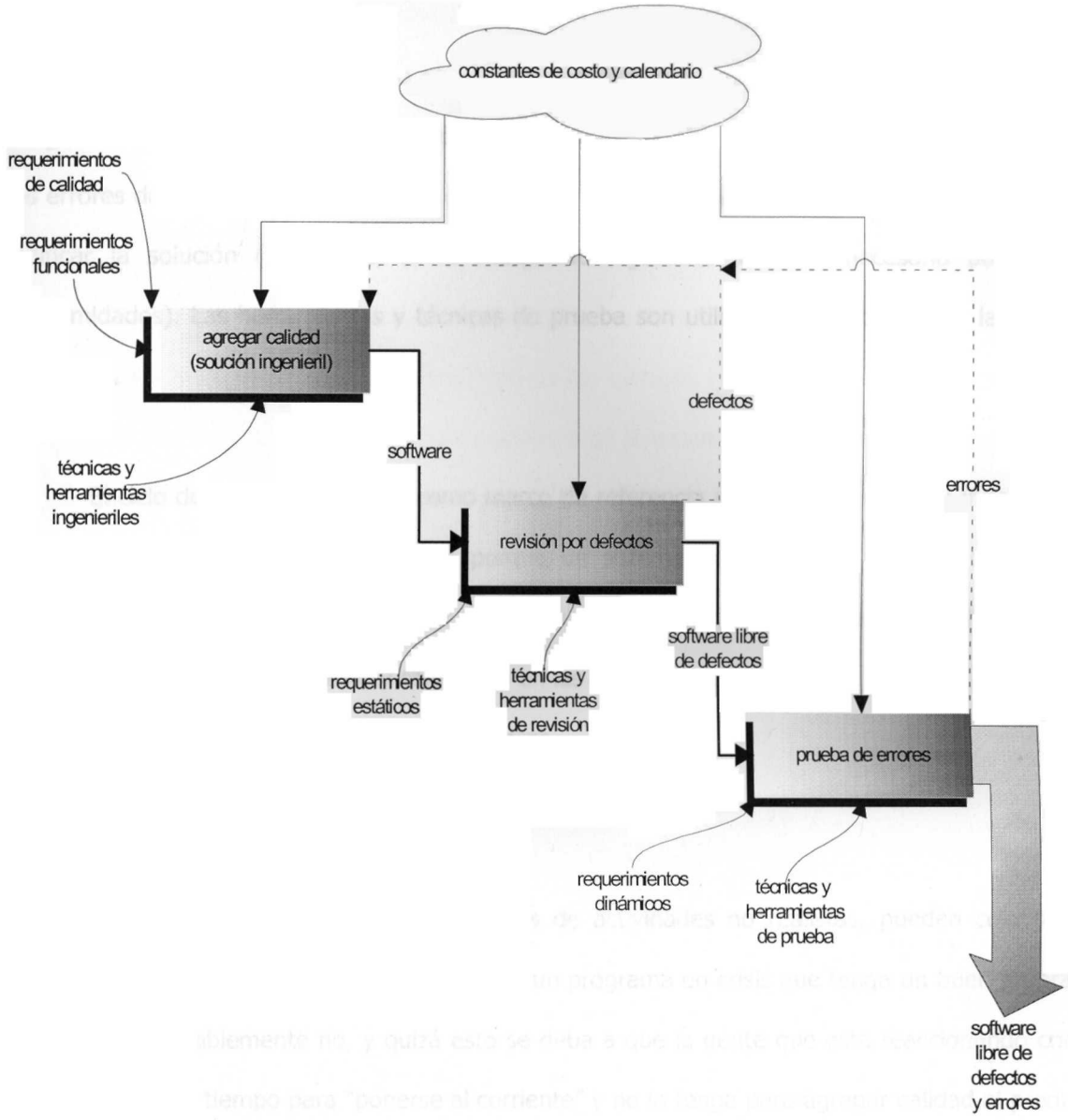


Figura 2.2. El ciclo de vida de la calidad del software.

El segundo paso en el ciclo de vida es revisar el diseño y el código. Esta actividad transforma los diseños y códigos del software en dos productos: defectos de código y software libre de defectos. Los defectos de diseño y código se convierten en controles en la actividad de ingeniería para re-diseñar o re-codificar la solución (esto es, el re-trabajo de ingeniería se hace necesario por las no-conformidades). Las herramientas y técnicas de revisión son utilizadas para completar la actividad de revisión.

La última actividad del ciclo es probar el código resultante. Esta actividad transforma el código libre de defectos en dos productos: errores de software y código libre de errores y defectos. Los errores de código se convierten en controles en la actividad de ingeniería para re-diseñar o re-codificar la solución (de nuevo, el re-trabajo de ingeniería se hace necesario por las no-conformidades). Las herramientas y técnicas de prueba son utilizadas para completar la actividad de prueba.

El ciclo de vida no solo sirve como marco de referencia para discutir sobre la ingeniería en la calidad del software, también indica porque un sistema de calidad es o no eficiente. Tres controles que pueden llegar a ser cuellos de botella en la eficiencia de un sistema de calidad son:

1. Controles de presupuesto y calendario en las actividades.
2. Efectividad de las técnicas y herramientas.
3. Volumen de re-trabajos.

Presupuestos inadecuados y programas de actividades no realistas, pueden colapsar un programa de calidad; o ¿se ha visto alguna vez un programa en crisis que tenga un buen programa de calidad?. Probablemente no, y quizá esto se deba a que la gente que esta reaccionando contra la crisis solo tiene tiempo para "ponerse al corriente" y no lo tenga para agregar calidad al producto o tiempo para hacer una revisión o tiempo para una prueba exhaustiva. Además las crisis cambian las prioridades administrativas: cumplir con las rutas críticas y actividades clave son las metas más importantes, mientras que la calidad se vuelve un elemento que se puede "omitir".

Las técnicas y herramientas son mecanismos para completar las actividades de ingeniería, revisión y prueba. Las técnicas y herramientas de ingeniería han sido buenas si resultan en un alto porcentaje de los atributos de calidad requeridos. Las técnicas y herramientas de revisión han sido buenas si éstas nos llevan a descubrir un alto porcentaje de los defectos. Las técnicas y

herramientas de prueba han sido buenas si éstas nos llevan a descubrir un alto porcentaje de los errores.

El volumen del flujo que se da en las retroalimentaciones de defectos (revisión) y errores (prueba) es un paso crítico en a eficiencia del sistema de calidad. Obviamente, entre más cargado sea el tráfico en estos ciclos de retroalimentación, se necesitan más retrabajos y en consecuencia, baja la eficiencia en el sistema. La efectividad de las técnicas y herramientas de ingeniería, junto con otros factores como habilidad personal, políticas de la organización y disponibilidad de recursos son los mayores controles de este flujo.

CAPITULO III

PRUEBAS EN EL SOFTWARE

El desarrollo del software se divide en varias fases, descritas en la literatura de pruebas de software y que esta descrita brevemente en el apartado C de este capítulo. Las pruebas del software cubren todas las etapas, comenzando por la definición de los casos de prueba del sistema y hasta su liberación, todo el proceso tiene su equivalente en las pruebas.

Ahora, para definir correctamente el concepto de la prueba del software es necesario entender primero algunos conceptos como: ¿qué es una anomalía? ¿Cuál es el propósito de las pruebas? ¿Cuáles son las metas de las pruebas? ¿Cómo las pruebas pueden ser divididas en módulos, fáciles de manejar? ¿Qué herramientas pueden ser usadas? y ¿cómo las pruebas pueden estar asociadas con el aseguramiento de la calidad del software (SQA)?.

1. Definición: error de software

En el estándar ANSI/IEEE 1044-1993, error se define como sigue: una anomalía (error) es cualquier condición que sale de lo esperado. Esta expectativa puede venir de la documentación (especificaciones de los requerimientos, diseño de documentos, documentos del usuario, estándares, etc.) o de las opiniones o experiencias de alguien. Una anomalía no es necesariamente un problema en el producto de software; éste puede trabajar de manera correcta por lo que un cambio en el software sería una mejora. Una anomalía puede ser causada también por otra cosa que no sea el software, por ejemplo memoria RAM en mal estado, dispositivos de entrada dañados, periféricos con fallas eléctricas, etc.

Pocos escritores del tema de pruebas en el software tienen sus propias definiciones para un error de software. Según Hetzel, un error de software es la acción humana que resulta en un software que contiene un incidente que, si se presenta, puede causar una falla. Según Kaner, una

discordancia entre el programa y su especificación es un error en el programa si y solamente si la especificación existe y es correcta.

Un error tiene muchos nombres: falla, incidente, descompostura, defecto, problema, queja, interferencia, defecto, fallo de funcionamiento y anomalía, aunque ahora se debe incluir el caló de la cultura informática: bug. La rutina, procedimiento, algoritmo, etc. del software donde existe una anomalía puede ser distinguida con alguno de los anteriores nombres.

Los errores del software se clasifican en diversas categorías. Esta clasificación cambia dependiendo de la literatura. Las anomalías se clasifican en ANSI/IEEE estándar 1044-1993: clasificaciones estándares para las anomalías del software. Las categorías de Beizer y Kaner son algo diferentes, pero ambas son igual de útiles.

En el estándar ANSI/IEEE 1044-1993 los errores se dividen en ocho clases principales:

1. problema de lógica
2. problema de cómputo
3. problema de interfaces/sincronía
4. problema del tratamiento de datos
5. problema de los datos
6. problema de la documentación
7. problema de la calidad del documento
8. mejoras o realce

Cada una de estas clases se divide en subclases que a su vez definen el error en cuestión de forma detallada. Por ejemplo, la clase del problema de la calidad del documento se divide en cinco clases de errores: Estándares de aplicación no satisfechos, no detectados, no actuales, inconsistencias e incompleto.

2. Definición: prueba de software

La definición de prueba es muy significativa. Dependiendo de la definición, el objetivo de la prueba se fija de diferente manera. Algunas definiciones contienen objetivos que son imposibles de lograr y algunas son psicológicamente insatisfactorias. Es importante encontrar la definición correcta que conduzca la prueba de forma más eficiente logrando resultados óptimos; que motive a la gente que está haciendo el trabajo de comprobación, de tal manera que sean más inquisitivos.

Por ejemplo, la definición de que prueba es un proceso para demostrar que no hay errores presentes, es "tanto teórica como prácticamente imposible"[3]. Esto es verdad virtualmente para todos los programas, aun para los triviales.

```
contador = 0
do {
        if ( a < b ) {
                switch( c ) {
                case 1: ...
                case 2: ...
                case 3: ...
                        case 4: ...
                }
        }
        contador ++;
} while ( contador < 20 );
```

Figura 3.1. Ejemplo de programa trivial.

[3] Beizer, Boris. Software System Testing and Quality Assurance, Van Nostrand Reinhold Co., New York, 1984, pp. 4.

Primero, la figura 3.1 contiene un ejemplo de un programa trivial que incluye un ciclo (contador del ciclo a partir del 0 al 20), una declaración if y una estructura case dentro de este ciclo. Para ejecutar este programa satisfaciendo los requisitos del tipo de prueba "exhaustiva" (significa que cada valor posible que acepte el programa debe ser ejecutado), se crean semillas para ejecutar y monitorear cada posibilidad que puede tener el programa, esto podría requerir para un entero en lenguaje de programación "c":

$$C\left(\frac{65536}{3}\right)=\left(\frac{65536!}{3!\bullet[65536-3]!}\right)$$
$$=\left(\frac{65536!}{3!\bullet 65533!}\right)$$
$$=\left(\frac{1}{6}\right)(65536*65535*65534)$$
$$= 4.691034865664\text{e}+13 \text{ de pruebas}$$

Segundo, el software bajo prueba consiste normalmente en varias partes mucho más grandes de código que este programa de ejemplo. Por lo tanto, la forma de prueba exhaustiva, como la captura o alimentación de información exhaustiva, parece ser impráctico, sino es que hasta imposible.

Algunos límites teóricos a la prueba exhaustiva son los siguientes pensamientos de Manna: "nunca podemos estar seguros que las especificaciones están correctas", "ningún sistema de prueba pueden identificar todos lo errores y aplicarse a todos los programas" y "nunca podemos estar seguros de que un sistema de prueba es correcto"[4].

[4] Manna, Zohar; Waldinger, Richard. The Logic of Computer Programming. IEEE Transactions on Software Engineering SE-4, 1978, pp. 2.

Otra forma de definir prueba es demostrando que el programa trabaja como éste ha sido definido, es decir, éste no ejecutará ninguna otra función más que la especificada y no causará ningún efecto lateral no deseado.

Sin embargo, las dos definiciones anteriores de prueba son ineficaces e insuficientes. Tienden a probar que el software trabaja correctamente y no se prueba otra cosa que este sin especificar. El objetivo de probar que el software es libre de error puede subconscientemente dirigir pruebas hacia esta meta. La persona que prueba el software tiene la tendencia a seleccionar los datos que tienen una probabilidad baja de hacer que el programa falle. El mismo efecto psicológico se vuelve verdad cuando uno está probando su propio código. La prueba es un proceso destructivo y "un programador no desea hacer que falle su propio código"[5]. Esto se podría aplicar en un grupo más grande que desarrolla software, por eso no deben probar sus códigos después de cierta fase (ejemplo: prueba de módulo) en vez de hacerlo ellos, este trabajo debe ser realizado por los grupos independientes de reconocimiento.

Por lo tanto una definición más conveniente para prueba es, "el ejecutar un software o un sistema con la intención de encontrar la mayor cantidad de errores"[6]. Este es psicológicamente, un mejor objetivo y motiva una mejor exploración, además es una meta más práctica el encontrar tantos errores como sea posible en lugar de verificar que el programa trabaja correctamente.

Una adición importante a la definición de prueba es que el objetivo de la prueba es el encontrar errores, pero no la solución a los mismos (es decir, probar pero no corregir).

---

[5] Myers, G.J. The Art of Software Testing, John Wiley and Sons, New York, 1979, pp. 39.
[6] Hetzel, W.C. Program Test Methods, Prentic-Hall, New Jersey, 1972, pp. 12.

La definición de prueba según el estándar ANSI/IEEE 1059-1993 es que "la verificación es el proceso de analizar un software para detectar las diferencias entre las condiciones requeridas y existentes (que sean defectos/errores/bugs) y evaluar las características del software"[7].

El objetivo de la prueba y el número de los errores encontrados contra todos los errores que existen en el código, depende del software bajo verificación. Por ejemplo, los requisitos para el software médico y el de la industria aeronáutica son mucho más altos que los requisitos para el software usado en una escuela. Esto significa que el software al que no se le permite "comportarse de una manera incorrecta" y que no puede contener casi ningún error, necesita pruebas más exactas que el software de uso general. Si el mal comportamiento no es tan serio, ejemplo, en editores de textos, entonces los usuarios tolerarán algunos errores más fácilmente. Las organizaciones de desarrollo se han fijado límites razonables a los errores del software; éstos límites son influenciados por requisitos de clientes y por los estándares de calidad tales como ISO9001.

En la práctica, la experiencia en la industria del software nos ha demostrado que es imposible producir y liberar software **sin un solo error**. El propósito de las pruebas es el mejoramiento de la calidad del producto. Aunque cuando se programan sistemas diseñados por el cliente, calidad significa cumplir con las especificaciones que él nos señaló. Si al cliente no le gusta el resultado final, no le preocupa si el producto cumple con las especificaciones finales (aún cuando el usuario estaba de acuerdo con la especificación); para ese comprador no hay satisfacción en calidad.

Si pensamos que la calidad en la tecnología de software significa "cumplir con los requerimientos" entonces es fácil definir el concepto de prueba como la medida de la calidad del

---

[7] IEEE Standards Board,IEEE Guide for Software Verification and Validation Plans, The Institute of Electrical and Electronics Engineers, Inc., New York, 1993, pp. 2.

software. El aseguramiento de calidad del software contiene naturalmente (como ya se vio en el capítulo II) más partes que la prueba, tales partes como las revisiones, métodos, etc., incluyen el ciclo entero de desarrollo de producto. La prueba del módulo y del protocolo son partes apenas pequeñas pero importantes de la SQA.

3. Actividades de prueba durante el ciclo de desarrollo de software

En muchas organizaciones, las pruebas son consideras la última fase en el ciclo de desarrollo de software momentos antes de su liberación, causando que la aplicación de las pruebas tenga su propia etapa haciendo con esto que cuando una nueva falla es descubierta se tenga que realizar de nuevo todo el trabajo o al menos reestructurar los casos de prueba que han sido pobremente planeados, o fueron diseñados para otra situación. Esta es una manera incorrecta de pensar; la prueba se debe ver como actividad continua en todo el ciclo de desarrollo.

El ciclo de desarrollo de software esta dividido en varias pequeñas etapas fácilmente manejables. Las fases secuenciales a partir de la etapa de los requisitos a la parte de la aceptación, forman la entidad conocida como el ciclo de desarrollo de software.

El estándar ANSI/IEEE 1074-1991, estándar de IEEE para el ciclo de vida del desarrollo de software, no contiene ningún ciclo de desarrollo exacto; éste sólo contiene un modelo general que puede ser adaptado a toda organización que se dedique al desarrollo de software y se puede implementar de acuerdo a su propio ciclo de desarrollo, y que obviamente cubra todas sus necesidades.

Cuando dos maneras diferentes de describir el ciclo de desarrollo de software, un modelo de "V" y un modelo de la "cascada", se comparan desde el punto de vista de prueba, se puede notar la superioridad del modelo de "V" (figura 3.2). El modelo de la cascada para el desarrollo del software presenta a la prueba como la fase final del ciclo de desarrollo, donde esta ubicada

actualmente, en cambio, el modelo de "V" muestra mejor la correlación entre las fases de prueba y las fases de desarrollo.

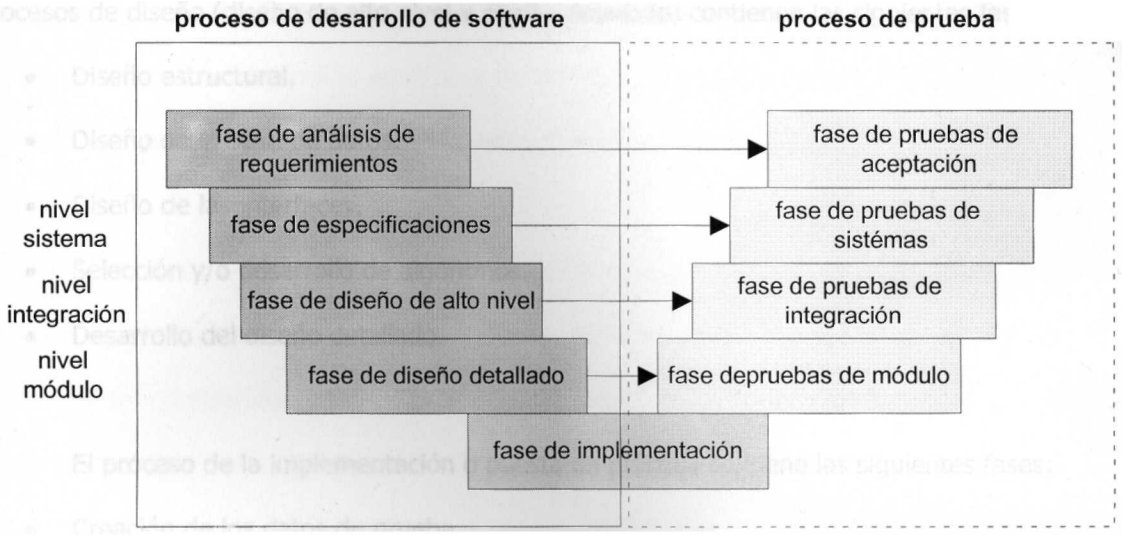


Figura 3.2. Modelo "V" del ciclo de desarrollo del software.

En este modelo, cada fase en el proceso de desarrollo de software (a la izquierda) tiene su fase correspondiente en el proceso de prueba (a la derecha). Por ejemplo, cuando se ha hecho el análisis de requerimientos, la planeación de los casos de prueba para verificar los requerimientos puede ser comenzada; a la ejecución de estos casos de prueba se le conoce como: prueba de aceptación. Algo muy significativo en el modelo es que la verificación y corrección de errores encontrados durante las mismas se pueden hacer en cualquier momento del ciclo de desarrollo. Sin embargo, el costo de encontrar y corregir los errores aumenta considerablemente según el nivel en que nos encontremos más abajo del lado izquierdo del modelo "V".

El análisis de requerimientos tiene las siguientes fases:

- Definición y desarrollo de los requisitos del software.
- Definición de los requisitos de la interfaz.
- Definición de las prioridades e integración del software.
- Requerimientos.

En la fase de especificaciones, se hacen las propuestas para el software previsto. Los procesos de diseño (diseño de alto nivel y diseño detallado) contienen las siguientes fases:

- Diseño estructural.
- Diseño de la base de datos.
- Diseño de las interfaces.
- Selección y/o desarrollo de algoritmos.
- Desarrollo del diseño detallado.

El proceso de la implementación o puesta en práctica contiene las siguientes fases:

- Creación de los datos de prueba.
- Creación del código fuente.
- Generación del código objeto.
- Creación de los manuales de operación.
- Planeación de la integración.
- Puesta en marcha de la integración.

En los libros de investigación de autores como Myer, Hetzel, Beizer y ANSI/IEEE definen como el punto de arranque para el ciclo de desarrollo del software, a la fase del análisis de requerimientos, entendiéndose como la definición de los requerimientos que un cliente tiene para el producto a ser desarrollado.

4. Taxonomía de las fallas

"Los gastos en las pruebas de un nuevo sistema son de aproximadamente el 50% del tiempo empleado y el 50% del costo total"[8].

---

[8] Myers, G.J. The Art of Software Testing, John Wiley and Sons, New York, 1979, pp. 1.

Es muy útil comenzar la fase de pruebas tan pronto como sea posible porque entre más pronto se encuentren las fallas de funcionamiento menos re-trabajos se harán en el futuro. Esto significa que si se encuentra una falla en la fase de prueba del módulo o de integración en vez de la fase de prueba del sistema o, en el peor de los casos, por el cliente, entonces cada etapa deberá ser vuelta a ejecutar después de que se haya cambiado el código.

El aumento de la cantidad de re-trabajos incrementa el costo de un producto de software. Los fallos encontrados en una última fase pueden también retrasar la agenda de actividades, lo que hace que los costos aumenten. Cada una de estas declaraciones comprueba, que es mejor comenzar a probar (aún en las primeras fases) tan pronto como sea posible.

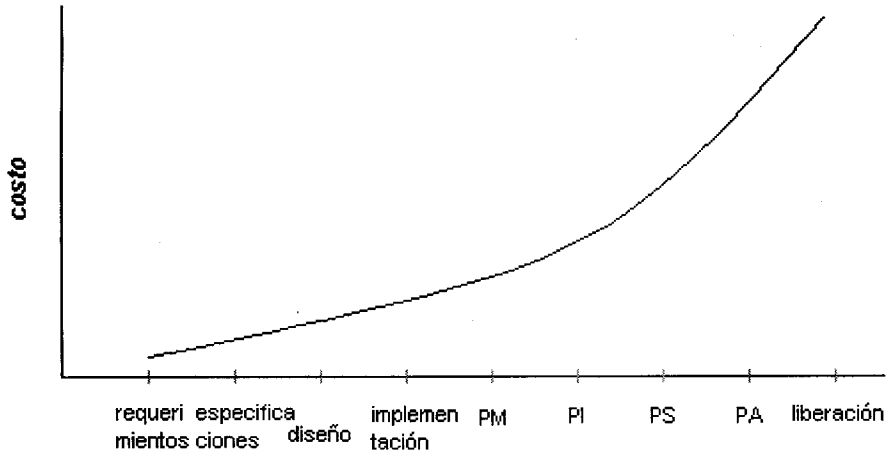


Figura 3.3. Costo de detectar y corregir problemas de software. Significado de las abreviaciones: PM – prueba de módulo, PI – prueba de integración, PS – prueba del sistema, PA – prueba de aceptación.

Un ejemplo del costo de los fallos es: "si el costo de detectar y de corregir un error en la fase de análisis de requerimientos es $10, los investigadores conservadores dan un costo de $100 para detectar y corregir el mismo error durante la prueba del proceso y $1,000 cuando el software esta listo para ser liberado"[9].

[9] Boehm, B.W. Software Engenniering, The Institute of Electrical and Electronics Engineers, Inc., New York, 1979, pp. 5.

Aunque la aplicación de pruebas en una fase temprana del ciclo de vida del software aumenta la cantidad de trabajo y por lo tanto los costos, estos costos son compensados por una disminución de los mismos en las últimas fases del ciclo de vida. Si los errores se han eliminado desde el comienzo, podemos suponer que las funciones del programa mejoran y que los costos para su mantenimiento serán más bajos.

5. Niveles de prueba

El proceso de prueba se divide en etapas más pequeñas, fácilmente manejables que se llaman niveles. Estos niveles son el componente más pequeño que al combinarlo o integrarlo con otros en unidades simples después formarán los componentes más grandes o fases. El estándar ANSI/IEEE 1059-1993 describe estos niveles de prueba como: **módulo** (componente), **integración, sistema**, y **aceptación.**

El proceso de prueba está estrechamente relacionado con el proceso de desarrollo del software. Esto significa que la verificación de cierta etapa es ejecutada usando un caso diseñado para la misma del proceso de desarrollo del software. El propósito de tal prueba es verificar cada parte del software contra sus definiciones. Cuando se encuentra un error, primero se busca su causa en el nivel precedente, y poco a poco el alcance de la búsqueda puede ser delimitada para finalmente llegar al nivel base del módulo, donde el error puede ser corregido. Cuando se corrige el error, se comprueba de nuevo el módulo actual comenzando por la prueba de módulo, y en este caso el propósito de la prueba es confirmar que ya no existe el error y que no se ha distorsionado creando nuevas fallas.

Según el estándar ANSI/IEEE 1059-1993, la prueba del **módulo** se define como "la prueba conducida para verificar la implementación del diseño para un elemento del software (ejemplo, una función o algoritmo) o una colección de elementos del software"[10]. El propósito de la prueba según el mismo estándar es asegurarse de que la lógica del programa es completa y correcta, además de asegurarse de que el componente trabaja según el diseño. Sin embargo esta clase de actitud puede causar una forma incorrecta de pensar. Por lo tanto, un mejor propósito de la verificación, de acuerdo con Myer, no es mostrar que el módulo corresponde a su especificación, sino mostrar que el módulo no contradice la especificación.

Según IEEE 1059-1993, la prueba de la **integración** se define como una progresión ordenada de pruebas en la cual los elementos del software se combinan y se prueban hasta que se ha integrado todo el sistema. El propósito de la prueba de integración, según el estándar de ANSI/IEEE, es asegurarse de que los objetivos de diseño han sido cumplidos. Sin embargo, según Beizer, "la prueba de la integración es una prueba hecha para demostrar que aunque los componentes eran individualmente satisfactorios, según lo demostrado al aceptar las pruebas de los componentes, la combinación de los componentes es incorrecta o inconsistente"[11]. Por ejemplo, para los componentes A y B, los cuales han pasado sus pruebas de módulo; la prueba de la integración tiene como objetivo el mostrar que no existe alguna inconsistencia entre A y B.

La prueba del **sistema** es el proceso de verificar un sistema integrado de hardware y de software para demostrar que el sistema cumple con sus especificaciones. El propósito de la prueba del sistema no es asegurarse de que el software (como entidad completa) cumple con los requisitos operacionales como se define en el estándar de ANSI/IEEE, pero si debe indicar cuando el sistema contradice los requisitos especificados.

---

[10] IEEE Standards Board,IEEE Guide for Software Verification and Validation Plans, The Institute of Electrical and Electronics Engineers, Inc., New York, 1993, pp. 3.
[11] Beizer, Boris. Software System Testing and Quality Assurance, Van Nostrand Reinhold Co., New York, 1984, pp. 83.

Diseñar buenos casos de prueba para sistemas requiere de por lo menos de tanta creatividad, inteligencia y experiencia como fue requerida para diseñar el software mismo. Myers propone que las siguientes 15 categorías se deben explorar al diseñar los casos de prueba:

1. Pruebas de recursos
2. Pruebas de volumen
3. Pruebas de tensión
4. Pruebas de utilidad
5. Pruebas de seguridad
6. Pruebas de funcionamiento
7. Pruebas de almacenamiento
8. Pruebas de configuración
9. Pruebas de compatibilidad/conversión
10. Pruebas de inestabilidad
11. Pruebas de confiabilidad
12. Pruebas de recuperación
13. Pruebas de utilidad
14. Pruebas de documentación
15. Prueba del procedimiento

La definición de la prueba de **aceptación** según el estándar ANSI/IEEE-1059 es: "prueba formal conducida para determinar si un sistema satisface o no sus criterios de la aceptación y permite al cliente determinar si acepta o no el sistema"[12]. El propósito de la prueba de aceptación es asegurarse de que los objetivos de los requisitos de los clientes están resueltos y de que todos los componentes están incluidos correctamente en un todo.

---

[12] IEEE Standards Board,IEEE Guide for Software Verification and Validation Plans, The IEEE; Inc., New York, 1993, pp. 3.

6. Documentos de prueba

El ciclo de vida del desarrollo es caracterizado por los documentos elaborados en cada fase del trabajo (estas fases están descritas en la figura 3.2) de la misma manera, el ciclo de vida de la prueba puede ser definido y controlado por los documentos generados en cada una de las actividades de prueba. Esta relación se muestra en la figura 3.4.

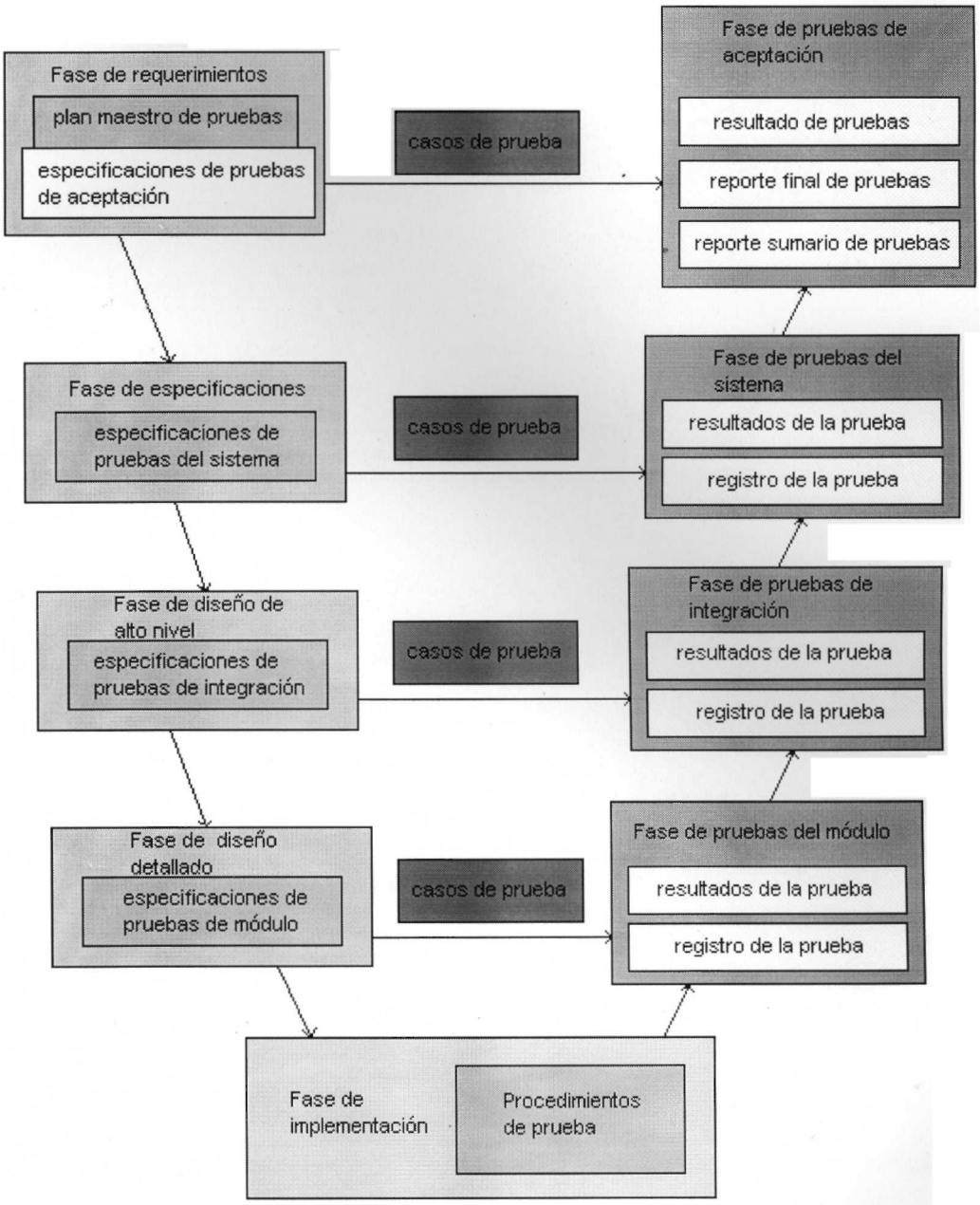
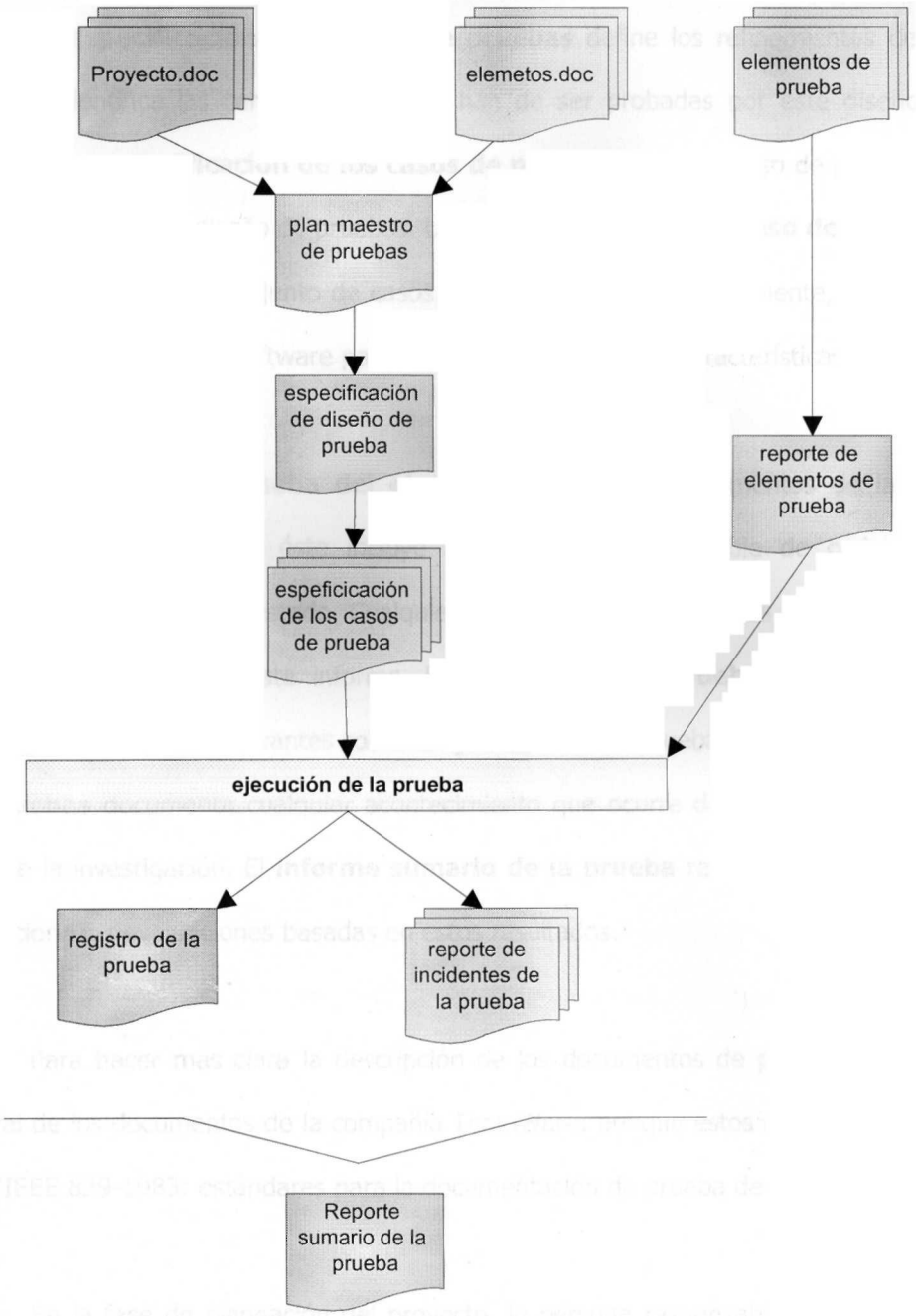


Figura 3.4. Documentos de prueba en el modelo "V" del ciclo de desarrollo de software.

La IEEE Computer Society Software Engineering Satandard Committee ha homologado el área de la documentación de pruebas y han producido una serie de estándares que fueron adoptados por ANSI (el mismo estándar ANSI es un ejemplo de un conjunto de documentos de pruebas). El estándar ANSI/IEEE 829-1983, define el propósito, objetivo y contenido de cada documento base. Esta y la relación de los documentos de las pruebas con los procesos de evaluación se describen en la figura 3.5.

Figura 3.5. Relación de los documentos de prueba con el proceso de prueba.



El propósito del plan de prueba es prescribir el alcance, enfoque, recursos y el calendario de las actividades de prueba e identificar los elementos a ser probados, las características que se probarán, las actividades de prueba a ser aplicadas, el personal responsable de cada tarea y los riesgos asociados a este plan.

La **especificación del diseño de pruebas** define los refinamientos del enfoque de la prueba e identifica las características que han de ser probadas por este diseño y sus pruebas asociadas. La **especificación de los casos de prueba** define un caso de prueba identificado por una especificación de diseño de pruebas. La **especificación de proceso de prueba** determina los pasos para ejecutar un conjunto de casos de prueba o, más generalmente, los pasos usados para analizar un elemento del software para evaluar un conjunto de características.

El **reporte de prueba del elemento** identifica los elementos de la prueba que son transmitidos para evaluar; éste incluye a la persona responsable de cada elemento, de su localización física, y de su estado. Cualquier variación en los requisitos actuales del elemento y los diseños se registran en este informe. El **registro de la prueba** proporciona un expediente cronológico de detalles relevantes sobre la ejecución de las pruebas. El **informe de incidentes de las pruebas** documenta cualquier acontecimiento que ocurre durante el proceso de prueba que requiere la investigación. El **informe sumario de la prueba** resume el resultado de la prueba y proporciona las evaluaciones basadas en estos resultados.

Para hacer más clara la descripción de los documentos de prueba se hará una mención general de los documentos de la compañía TeamWare; aunque éstos difieran un poco del estándar ANSI/IEEE 829-1983: estándares para la documentación de prueba de software.

En la fase de planeación del proyecto, la persona responsable de las pruebas elabora un plan maestro de pruebas general que abarca la prueba del producto o del sistema o ambos. Los diseñadores de los casos reciben toda la información necesaria de la documentación del plan de proyecto (ejemplo: especificaciones de los requerimientos, especificaciones funcionales y documentación del plan del proyecto).

El plan maestro de prueba es un documento que incluye los recursos que se utilizarán y un calendario para las actividades de prueba e identifica los elementos a ser probados como se definen en el estándar ANSI/IEEE 829-1983. Más aún, el plan especifica las herramientas de prueba, las condiciones necesarias y describe los procedimientos de validación así como el de manejo de los errores.

Una especificación detallada de la prueba es realizada al inicio de la fase de prueba. La configuración de prueba, por ejemplo, da una descripción más detallada del entorno de prueba. La especificación de prueba es un plan de prueba detallado para un producto o facilidad en específico como el correo, directorio y calendario de TeamWare; por ejemplo, en el se define cuál caso de prueba de la colección es seleccionado para la prueba actual y divide los detalles de la arquitectura de prueba en partes menores para hacer posible el entrar en detalle a la prueba.

La prueba utiliza métodos formales para registrar los errores, para reportarlos a los diseñadores, para validar y registrar las correcciones utilizando un sistema observador de problemas llamado NIPO (fabricado por TeamWare). Como resultado de la prueba se obtiene un reporte de los eventos que indica qué fue probado y que no, cuáles fueron los resultados; además incluye recomendaciones sobre modificaciones deseables y si el producto es aprobado o no. El reporte final de la prueba incluye todos los errores del sistema NIPO y los resultados detallados de las pruebas.

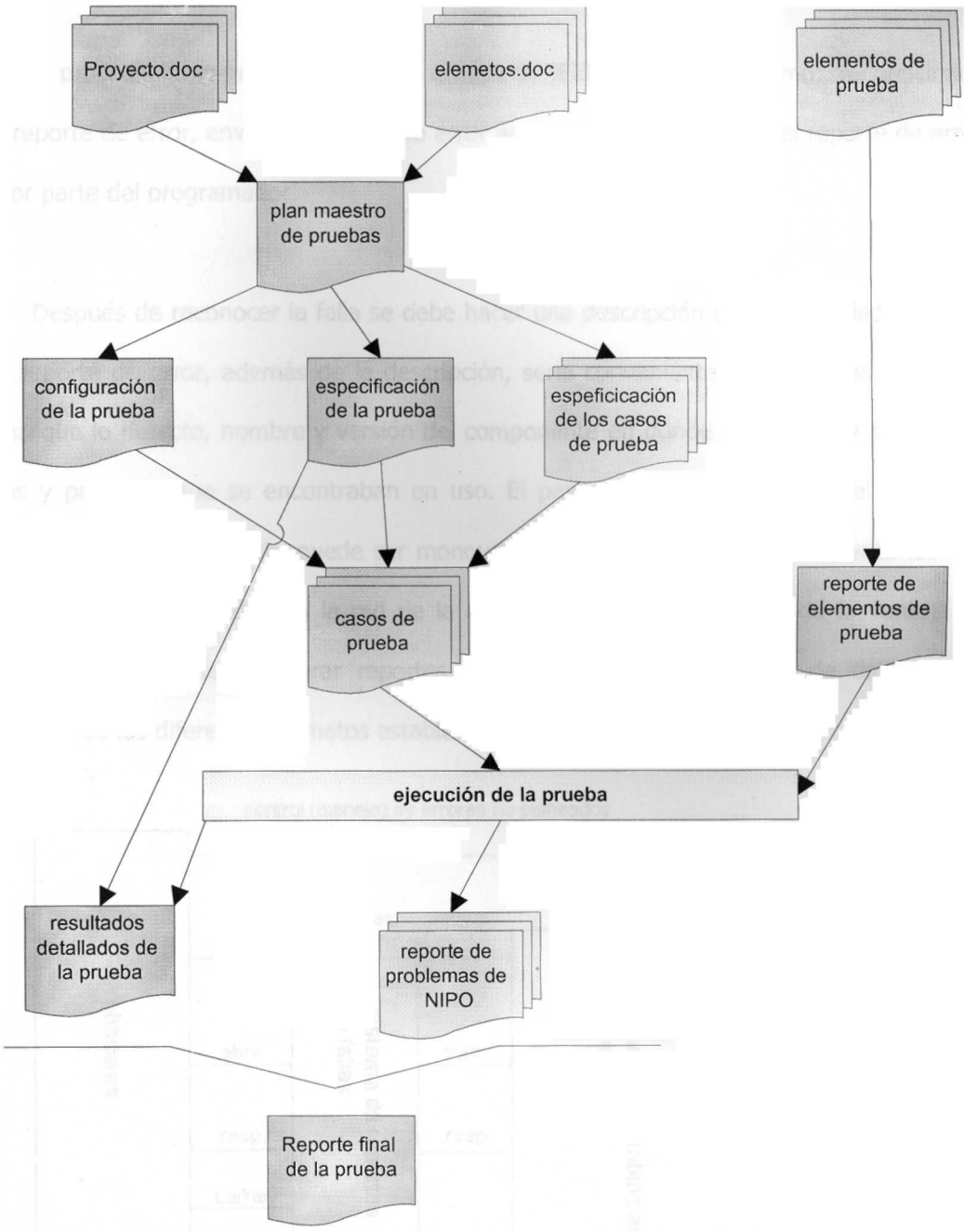


Figura 3.6. Prueba de documentos utilizando la estructura interna de TeamWare.

7. Procedimiento para el manejo de errores

El procedimiento para el manejo de errores es una de las partes esenciales de la detección y corrección de fallas. Estos ayudan en la diaria comunicación entre las organizaciones de prueba y las organizaciones fabricantes de software.

El procedimiento para el manejo de errores de IEEE incluye fases como: reconocimiento de error, reporte de error, envío del reporte de error al programador y recibo del reporte de arreglo de falla por parte del programador.

Después de reconocer la falla se debe hacer una descripción clara y detallada de la misma en un reporte de error, además de la descripción, sería conveniente incluir: fecha, nombre de la persona que lo detecto, nombre y versión del componente en dónde se detecto y situación de los campos y procesos que se encontraban en uso. El paso siguiente es capturar el reporte en un sistema de seguimiento; éste puede ser monousuario o multiusuario; generalmente los sistemas multiusuario están colocados en la red de la organización, lo anterior nos da la ventaja de que muchos usuarios pueden capturar reportes, realizar consultas a la base de datos y obtener información de los diferentes formatos establecidos.

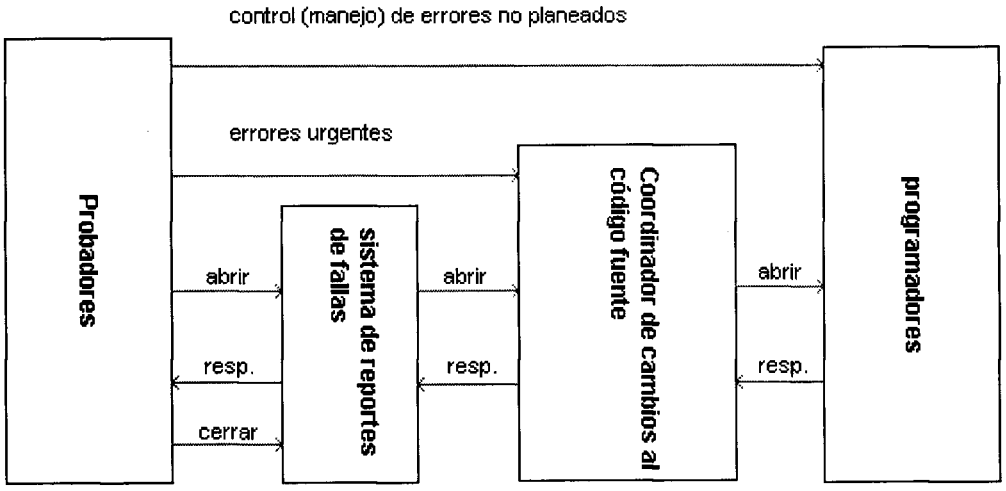


Figura 3.7. Manejo de errores utilizando un sistema de seguimiento.

Después que el reporte ha sido escrito, este **debe ser** contestado. Esta labor **debe ser** la tarea del programador una vez que se ha detectado, corregido, o en su defecto revisado, la falla. Este reporte debe indicar si la falla ha sido corregida para esa versión o si la corrección esperará hasta la siguiente. Cuando se resuelve el problema, el programador debe indicarlo en el sistema de

seguimiento, además de anotar claramente cuales fueron los pasos que realizó para solucionarlo; posteriormente debe avisar al grupo de pruebas que comiencen de nuevo con la verificación del programa.

El sistema de seguimiento debe asegurar que no se olviden las experiencias y soluciones a problemas que se han presentado; por lo que cuando se planea una nueva versión de software, los archivos históricos (base de conocimiento) son de gran importancia a la compañía y más que nada a los programadores de sistemas y equipos de pruebas.

CAPITULO IV

APRENDIZAJE ORGANIZACIONAL

1. ¿Qué es el aprendizaje organizacional?

La mayoría de las empresas humanas dan por hecho que el aprendizaje organizacional se relaciona de alguna manera con la enseñanza formal. Pero aprender implica mucho más que estudiar y el aprendizaje organizacional resulta mucho más complejo que el individual.

La mayor parte del aprendizaje organizacional tiene lugar en una serie de momentos aislados que los empleados experimentan a diario: contemplar las actividades en silencio, interactuar con las personas dentro o fuera de la organización, participar en el trabajo de grupos pequeños, leer documentos internos, desempeñar tareas, observar cómo se hace el trabajo.

Una definición sencilla de aprendizaje organizacional es: "averiguar qué da buenos resultados o qué da mejores resultados"[13]. Ahora, una definición más elaborada es: "adquirir y aplicar los conocimientos, técnicas, valores, creencias y actitudes que incrementan la conservación, el crecimiento y el progreso de la organización"[14].

Cabe resaltar un comentario muy oportuno sobre la aplicación del aprendizaje: el aprendizaje no es del todo completo si no se aplica de manera efectiva.

---

[13] Guns, Bob, Aprendizaje Organizacional: Como Ganar y Mantener la Competitividad, Prentice-Hall, México, 1996, pp. 16.
[14] Ob.cit. pp. 16.

2. ¿Qué es una organización aprendiente?

Existen varias definiciones de organizaciones aprendientes:

1. "Cualquier empresa enfocada al mejoramiento continuo de sus procesos, productos y servicios; que facilita el aprendizaje de sus miembros tanto de forma individual/independiente como en grupos/equipos; que se transforma continuamente para lograr sus metas"[15].
2. "De manera análoga al aprendizaje individual que se mide en la adquisición y desarrollo de habilidades, el aprendizaje organizacional se define como el incremento en las habilidades de la empresa para responder de manera efectiva a las situaciones que sean cotidianas"[16]. Entendiendo como cotidianas a aquellas que se presenten en el trabajo u ocupación.
3. "Organizaciones en las que las gentes expanden sus capacidades de manera continua para dar los resultados deseados, cuidando el nuevo pensamiento y los nuevos patrones, donde no hay restricciones a la inspiración individual y colectiva; donde la gente aprende continuamente cómo aprender juntos"[17].
4. "El aprendizaje organizacional es el resultado de la unión de tres esferas de actividad: aprendizaje individual, de equipo y sistémico. Los tres tipos de aprendizaje se efectúan de manera simultanea. El aprendizaje individual se lleva a cabo cada vez que un individuo lee un libro, realiza un experimento u obtiene retroalimentación de sus colegas o un equipo de trabajo. El aprendizaje de equipo se lleva a cabo cuando dos o más individuos de manera conjunta aprenden de la misma experiencia o actividad. El aprendizaje de equipo puede llevar implícito el hecho de asignar responsabilidades nuevas y la interacción entre los miembros del equipo. El aprendizaje sistémico se lleva a cabo cuando la organización desarrolla procesos sistémicos para adquirir, utilizar y comunicar conocimiento organizacional"[18].

---

[15] Denton, D.K. The Learning Organization Involves the entire work force, Quality Progress, Dec. 1992, pp.70.
[16] Flood, A.L., The Learning Organization, The worklife report, 1993, pp.2
[17] Stata, Ray, Organizational Learning, Sloan maagement review, Nov.-Dec. 1992, pp.97.
[18] Senge, Peter, La Quinta Disciplina: Cómo Impulsar el Aprendizaje en la Organizacion Inteligente, Vergara, Argentina, 1990, pp.3.

Una organización aprendiente envuelve una actitud y una atmósfera. El deseo de aprender puede basarse en personas, equipos, procesos, sistemas y estructuras. El aprendizaje es el valor central de la cultura de la organización. En este medio, la innovación no sólo es enfrentar sino también celebrar.

3. ¿Qué es aprendizaje?

Una definición del diccionario nos indica que el aprendizaje es: "adquisición de conocimientos o habilidades". Ambas partes de la definición son importantes: lo que aprende la gente (know how) y cómo lo entienden y aplican (know why); entonces el aprendizaje se puede definir como el incremento de una capacidad para realizar una acción de manera más efectiva.

Existen dos tipos de conocimiento:

- **Formal**, es el conocimiento que se encuentra registrado en libros, medios electrónicos, etc. y que puede ser consultado por cualquiera. Tiene la característica de que se conserva y es reproducible.
- **Tácito**, es el conocimiento que los individuos traen consigo mismos, no esta registrado en ningún medio físico reproducible y que termina en el momento en que el individuo desaparece o lo olvida.

Además debemos recordar que todo aprendizaje requiere de una reflexión, veamos el círculo del aprendizaje:

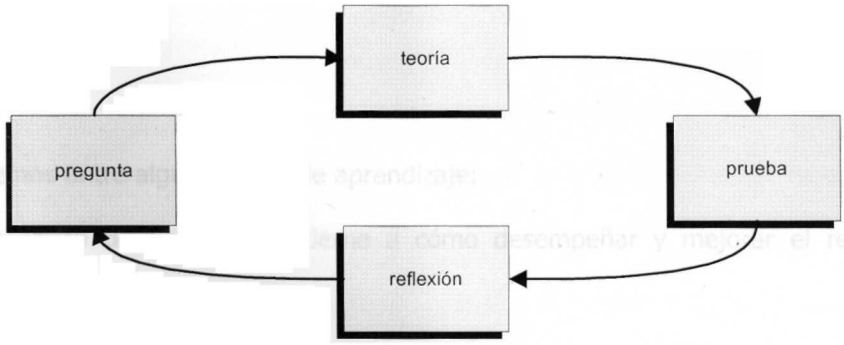


Figura 4.1. Círculo del aprendizaje.

Ray Stata comenta sobre el aprendizaje que: "All in the organization must master the cicle of thinking, doing, evaluating, and reflecting. Without, there is no valid learning"[19].

4. Tipos de aprendizaje

Hasta ahora se ha hablado del aprendizaje. Los empleados de las organizaciones también necesitan saber algo acerca del contenido del aprendizaje, es decir, el qué (objeto) al cual se aplica el proceso o conocimiento.

Cualquier negocio depende de cierto nivel mínimo de competencia en el aprendizaje de tareas. Frecuentemente, el sólo aprendizaje de las tareas puede determinar la posición competitiva. Si en una empresa y la de sus competidores se ha alcanzado casi el mismo nivel de aprendizaje de tareas, para poder obtener una ventaja competitiva sobre los demás, todos los tipos de aprendizaje se volverán importantes.

Un aprendizaje más rápido depende de dos factores:

1. Con qué rapidez aprenden los individuos y los grupos

[19] Stata, Ray, Organizational Learning, Sloan maagement review, Nov.-Dec. 1992, pp.97.

2. Con qué rapidez se transfiere el aprendizaje a otros individuos y grupos dentro de la organización.

Entonces, tenemos entre algunos tipos de aprendizaje:

- **Aprendizaje de tareas:** concierne a cómo desempeñar y mejorar el rendimiento de tareas específicas.
- **Aprendizaje sistemático:** tiene que ver con la comprensión de los sistemas y los procesos básicos de la organización, con la forma en la que se realizan y se ponen en práctica y con la forma en la que se pueden mejorar.
- **Aprendizaje cultural:** se centra alrededor de los valores, creencias y actitudes que proporcionan la base para un trabajo productivo.
- **Aprendizaje del liderazgo:** se concentra en la forma de guiar y controlar a los individuos, grupos, equipos y a las unidades organizacionales más grandes.
- **Aprendizaje de equipo:** tiene que ver con la forma de funcionar efectivamente en un equipo y fomentar su aprendizaje, crecimiento y madurez.
- **Aprendizaje estratégico:** se centra alrededor de la estrategia básica de negocios de la empresa, del modo en que se realiza y se pone en práctica y de la forma en la que se puede mejorar.
- **Aprendizaje empresarial:** concierne a los aspectos empresariales básicos y a la forma de dirigir a los equipos como micronegocios.
- **Aprendizaje reflexivo:** tiene que ver con cuestionar y analizar las hipótesis, los modelos y los paradigmas organizacionales.
- **Aprendizaje de la transformación:** se concentra en la forma de lograr un cambio organizacional significativo.

5. Niveles de aprendizaje

Existen cinco niveles en el aprendizaje:

1. **Adquisición.** El primer nivel consiste en adquirir actitudes, creencias, valores, principios, información, conocimiento y oficio. Gran parte de la adquisición tiene lugar incluso antes de contratar a un empleado.
2. **Utilización.** El segundo nivel consiste en utilizar los elementos adquiridos. Sin embargo, la utilización es sólo una actividad, no un aprendizaje real, a menos que se cree un círculo de retroalimentación de manera que el rendimiento real se pueda comparar con el rendimiento pretendido. Por ejemplo, una experiencia personal en la que yo decidí cambiar mi conducta basándome en una meta deseada (el rendimiento pretendido) y en la retroalimentación acerca de la forma en la que mi conducta anterior me impidió alcanzar una meta similar. El trabajo llega a depender de la capacidad de comprender la información, responder a ella, controlarla y crear un valor a partir de ella. Por consiguiente, las operaciones eficientes en el ambiente de trabajo informado requieren una distribución más equitativa del conocimiento y de la autoridad. La transformación de la información en riqueza significa que es necesario concederles a más miembros de la empresa las oportunidades de saber más y de hacer más.
3. **Reflexión.** El tercer nivel requiere el alejamiento del proceso, con el fin de "ver el bosque en vez de los árboles". La reflexión es pensar en la "perspectiva más amplia". La reflexión está libre de una acción externa. Se caracteriza por el interrogatorio, el análisis y la superación de suposiciones. Por ejemplo, un individuo aprendiente o un grupo que sepa reflexionar podría enfocarse en los aspectos culturales dentro de la organización y en el efecto que causan estos aspectos en la forma en la que está situado el negocio, o debería estarlo, en comparación con la competencia. Si vamos un poco más lejos, la reflexión podría implicar la construcción de nuevos paradigmas, o sea de modelos mentales de cómo funcionan las cosas. La reflexión en dichos paradigmas podría significar una nueva

definición de en qué negocio se encuentra usted y de la forma en la que realiza sus negocios.

4. **Cambio.** El cuarto nivel combina el pensamiento y la acción. La persona o el grupo responden a una oportunidad o un problema mediante una estrategia, asignando recursos y emprendiendo una acción con el fin de asegurarse de que el cambio deseado resulte en una aplicación de alto impacto del aprendizaje.
5. **Flujo.** Este nivel recibió su nombre del libro de Mihaly Csikszentmihalyi, titulado Flow (flujo). En el nivel del flujo, los aprendizajes mínimos se siguen reforzando unos a otros sin esfuerzo consciente. El aprendizaje y la actividad relacionada parecen unirse en una corriente que sigue su curso, hacia delante.

6. How... Enfoque de las cinco disciplinas para una organización aprendiente

Peter Senge en su libro la quinta disciplina plantea una serie de estrategias para lograr lo que él llama una organización aprendiente a partir de cualquier empresa humana. Mucho de su pensamiento hasta el momento se centra en un liderazgo eficiente, abierto, franco y motivacional. Senge comenta que no existe ninguna "receta de cocina" para lograr que una organización sea aprendiente, pero que el cambio de actitud que se puede tener al entender y aplicar (¿qué esto no es aprendizaje?) los conceptos de las cinco disciplinas, pueden hacer que se logre una empresa aprendiente que, en el corto plazo fundamente en éste pensamiento su ventaja competitiva del siguiente milenio. Cada una de las disciplinas desarrolladas tienen tres niveles que se pueden aplicar de manera personal o en equipo:

1. **Prácticas:** se relacionan con el qué hacer, son las actividades en que los practicantes de la disciplina concentran el tiempo y las energías.
2. **Principios:** ligados a las ideas rectoras y conceptos, éstos son tan importantes para el principiante como para el maestro. Para el principiante constituyen una ayuda para comprender la justificación de la disciplina y sus prácticas. Para el maestro, constituyen

puntos de referencia que lo asisten para refinar continuamente la practica de la disciplina y para explicarla a otros.

3. **Esencias:** el estado de ser de quienes tienen un gran dominio de la disciplina. Aunque son difíciles de expresar en palabras, son vitales para aprender plenamente el significado y propósito de cada disciplina.

Es casi imposible dominar una estrategia de manera inmediata y como es de esperar todas llevan consigo un proceso de aprendizaje. La presentación que aquí doy de los niveles hace suponer que, comenzando por la base, la adquisición de aptitudes nos llevara a la cúspide de la pirámide.

a. 1ª Disciplina: Dominio Personal

El sentido que se da a esta disciplina no es el de "dominar a los demás", sino el de tener una habilidad especial que permita aclarar y ahondar continuamente nuestra visión personal, concretar las energías, desarrollar paciencia y ver la realidad objetivamente, es decir, la disciplina del espíritu, del crecimiento y el aprendizaje personal. Es un proceso que dura toda la vida.

El dominio personal es lo que mantiene a los individuos comprometidos y motivados hacia su misión en la organización a través del tiempo, no importa que dejen de ser "jóvenes entusiastas". Desgraciadamente, pocas organizaciones lo alientan y después de algún tiempo, los individuos empiezan a perderlo y la organización entonces aprovecha muy poco de sus energías y casi nada de su espíritu. El hecho de querer dejar de aprender del "espíritu organizacional" depende del espíritu de sus integrantes, de la capacidad de la gente por aprender. Las organizaciones sólo aprenden a través de individuos que aprenden. La fuerza activa de una organización es la gente.

El dominio personal significa abordar la vida como una tarea creativa (proactiva) y no reactiva, lo cual es un esfuerzo permanente denominado "tensión creativa" por su símil con una liga que une (con tensión constante) nuestra realidad actual y nuestra visión del futuro.

Características de las personas con un alto nivel de dominio personal:

- Su visión es una vocación y no sólo una buena idea.
- Ven la "realidad actual" como un aliado, no como un enemigo.
- Aprenden a trabajar con las fuerzas del cambio en lugar de resistirlas
- Se sienten conectadas con otras personas y con la vida misma, sin perder su singularidad.

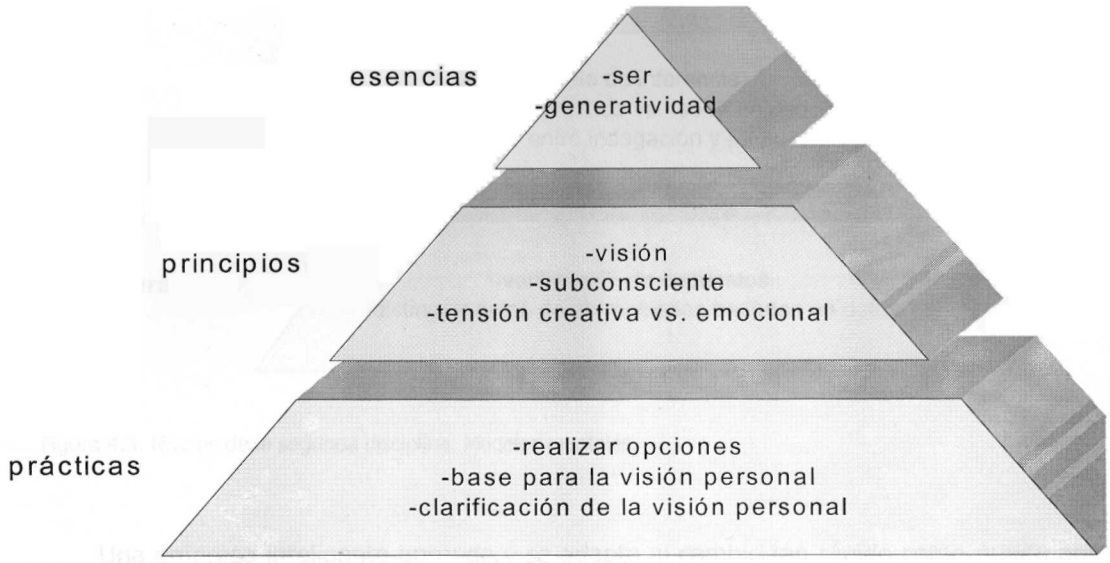


Figura 4.2. Niveles de la primera disciplina. Dominio personal.

- Se sienten parte de un proceso creativo más amplio en el cual pueden influir, pero no controlarlo unilateralmente.
- Viven en una continua modalidad de aprendizaje, nunca "llegan" de imprevisto.
- Son muy conscientes de su ignorancia, su incompetencia, sus zonas de crecimiento... y sienten una profunda confianza en sí mismos. ¿Paradójico?. Sólo para los que no ven que "la recompensa es el viaje".

- Realizan tareas extraordinariamente complejas con gracia y facilidad.
- Se concentran en el resultado deseado, no en el proceso.

b. 2ª Disciplina: Modelos Mentales

Se describe a los procesos mentales como supuestos hondamente arraigados, generalizaciones e imágenes que influyen sobre nuestro modo de comprender el mundo.

Figura 4.3. Niveles de la segunda disciplina. Modelos mentales.

Una empresa inteligente aprende y se adapta al cambio tan rápido como pueda aprender a cambiar y adaptar sus modelos mentales.

Yo no puedo esperar cambiar a los demás si no cambio primero; este es uno de los puntos importantes pero más difíciles de la psicología: la introspección. La persona a la que nos cuesta más trabajo comprender es a nosotros mismos, porque tenemos miedo de conocer cómo somos realmente. Sin embargo, este ejercicio es necesario porque las disciplina de trabajar con modelos mentales empieza por volver el espejo hacia adentro. Hacer brotar nuestras propias imágenes del mundo y someterlas a un riguroso escrutinio.

Decimos imágenes porque lo que llevamos dentro de la mente no es la realidad del mundo, ni la estructura de una organización, sino interpretaciones (imágenes, supuestos e historias) de ellas. Y la razón de que muchas buenas ideas no se lleven a cabo es porque chocan con esas interpretaciones internas profundamente arraigadas.

El problema con los modelos mentales no radica en que sean atinados o erróneos, sino en que sean tácitos, de tal forma que estén por debajo del nivel de conciencia y no nos demos cuenta de ellos. No se puede cambiar un modelo mental que no ha sido reconocido como tal.

c. 3ª Disciplina: Construcción de una visión compartida

Esta disciplina la define Senge como "la capacidad para compartir una imagen del futuro que se procura crear"[20]. Cuando hay una visión compartida genuina (no impuesta), la gente no sobresale ni aprende porque se lo ordenan, sino porque lo desea.

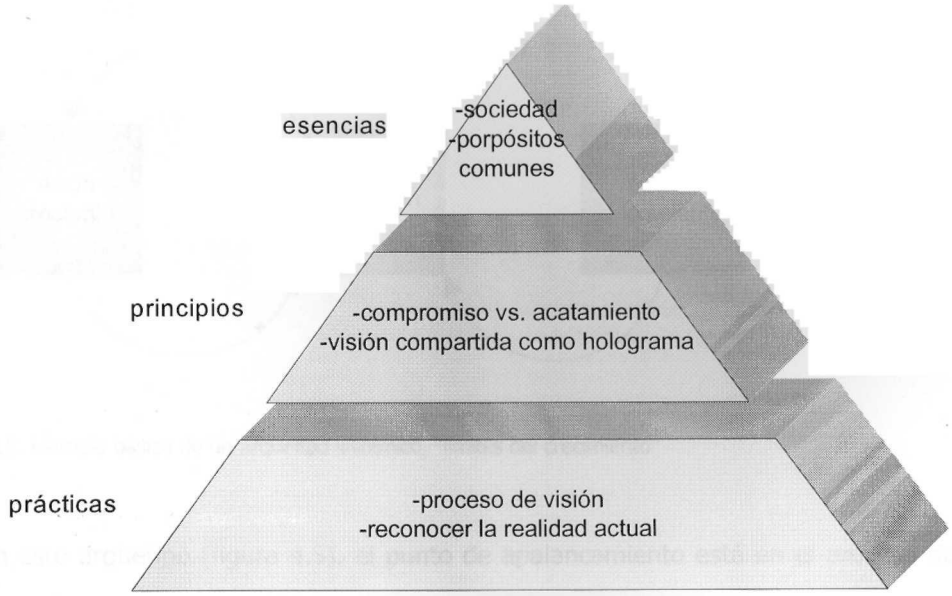


Figura 4.4. Niveles de la tercera disciplina. Construcción de la visión compartida.

[20] Senge, Peter, La Quinta Disciplina, Vergara, Argentina, 1989, pp.45.

Su práctica supone aptitudes para configurar "visiones del futuro" compartidas por compromiso genuino y no por acatamiento. La mayoría de las visiones actuales de las empresas son de una persona (o un grupo) y se imponen a la organización.

La visión compartida, en su nivel más simple, es la respuesta a la pregunta: ¿Qué deseamos crear?.

Las visiones compartidas surgen de visiones personales. De hecho, crecen como subproductos de interacciones de visiones individuales. La construcción de una visión compartida es sólo una parte de la actividad de crear las "ideas rectoras" de la organización: visión, misión y valores centrales.

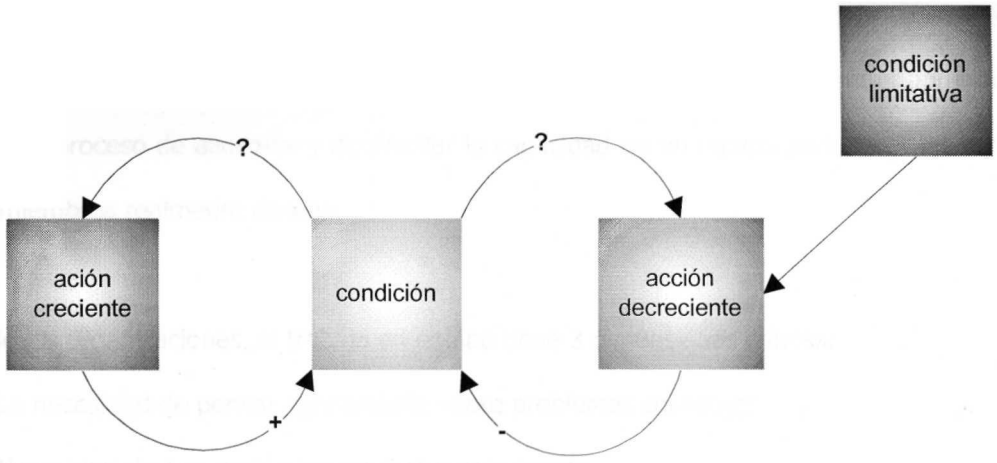


Figura 4.5. Ejemplo básico de un arquetipo sistémico, "límites del crecimiento"

En este arquetipo (figura 4.5), el punto de apalancamiento está en el proceso de balance. Ahí está la clave para conservar el ímpetu del proceso visionario de la empresa.

d. 4ª Disciplina: Aprendizaje en equipo

El aprendizaje en equipo es vital porque la unidad fundamental de aprendizaje en las organizaciones modernas no es el individuo, sino el equipo. Si los equipos no aprenden, la organización no puede aprender.

La inteligencia del equipo es superior a la de sus integrantes, desarrollando aptitudes extraordinarias para la acción coordinada. Cuando los equipos aprenden de verdad, no sólo generan resultados extraordinarios, sino que sus integrantes crecen con mayor rapidez.

Esta disciplina comienza con un dialogo abierto entre sus integrantes y su práctica supone un compromiso constante con el aprendizaje. El diálogo y la discusión son las dos maneras en que conversan los equipos. Ambas prácticas deben dominarse por los equipos.

El fenómeno principal del aprendizaje en equipo es el alineamiento, que consiste en que todos "empujen la carreta en el mismo sentido y hacia donde todos quieren", así el aprendizaje en equipo es el proceso de alinearse y desarrollar la capacidad de un equipo para crear los resultados que sus miembros realmente desean.

Dentro de las organizaciones, el trabajo en equipo tiene 3 dimensiones críticas:

1. La necesidad de pensar agudamente sobre problemas complejos.
2. Necesidad de una acción innovadora y coordinada.
3. El papel de los miembros del equipo en otros equipos.

e. 5ª Disciplina: Pensamiento sistémico

Esta disciplina es la que integra a todas las anteriores, fusionándolas en un cuerpo coherente de teoría y práctica. Sin una orientación sistémica no hay motivación para examinar el cómo están interrelacionadas las disciplinas.

El pensamiento sistémico permite comprender el aspecto más sutil de la organización inteligente, la nueva percepción que se tiene de sí mismo y el mundo. Permite ver películas y no sólo fotos instantáneas.

Figura 4.6. Niveles de la cuarta disciplina. Aprendizaje en equipo.

Figura 4.7. Niveles de la quinta disciplina. Pensamiento sistémico.

Pero también el pensamiento sistémico requiere de las otras disciplinas para realizar su potencial: la construcción de una visión compartida alienta un compromiso a largo plazo, los modelos mentales enfatizan la apertura necesaria para desnudar las limitaciones de nuestra manera actual de ver el mundo, el aprendizaje en equipo desarrolla las aptitudes de grupos de personas para buscar una figura más amplia que trascienda las perspectivas individuales, el dominio personal alienta la motivación personal para aprender continuamente.

f. Relación de las disciplinas primera y quinta

La perspectiva sistémica ilumina aspectos más sutiles del dominio personal, especialmente la integración de la intuición y la razón, compasión y compromiso con la totalidad. La intuición ha recibido creciente aceptación como una forma para la resolución de problemas. Siguen sus corazonadas. Las personas con altos niveles de dominio personal no se proponen integrar la razón con la intuición. Lo consiguen naturalmente. A medida que dominamos el pensamiento sistémico, descubrimos que muchas de nuestras intuiciones son explicables. Eventualmente, la reintegración de razón e intuición puede ser uno de los principales aportes del pensamiento sistémico.

g. Relación de las disciplinas segunda y quinta

Los modelos mentales arraigados frenan los cambios que podrían derivar del pensamiento sistémico. De hecho, las dos disciplinas van de la mano porque la de modelos mentales intenta exponer supuestos ocultos y la del pensamiento sistémico intenta reestructurar supuestos para revelar la causa de problemas cruciales.

El pensamiento sistémico también es importante para trabajar eficazmente con modelos mentales porque la mayoría de éstos tienen defectos sistemáticos al pasar por alto retroalimentaciones, juzgar erróneamente sus demoras involucradas y no concentrarse en puntos reales de apalancamiento (por estar ocultos muchas veces).

h. Relación de las disciplinas tercera y quinta

Un arquetipo asociado al sistema visionario es el de "límite de crecimiento". Hay visiones que jamás cobran arraigo porque siempre existe el ciclo de balance de diversidad de perspectivas, desaliento, etc. que frenan el ciclo reforzador del entusiasmo por la visión.

En síntesis, la disciplina de construir visiones compartidas, carece de un sustento crucial si se practica sin el pensamiento sistémico. La visión pinta la imagen de lo que deseamos crear. El pensamiento sistémico revela cómo hemos creado lo que tenemos ahora.

i. Relación de las disciplinas cuarta y quinta

Las herramientas del pensamiento sistémico son importantes porque casi todas las tareas de los equipos se enfrentan a una enorme complejidad, sobre todo los administrativos (estrategias, visiones, políticas, estructuras, etc.).

Los equipos requieren compartir un nuevo lenguaje para describir la complejidad. Los arquetipos sistémicos ofrecen una base potencialmente poderosa para un lenguaje que permitirá a los equipos abordar productivamente la complejidad. Sin este lenguaje, el aprendizaje en equipo es limitado.

¿Puede este cuerpo de conocimiento generar una teoría de apoyo para tener organizaciones más efectivas, proceso de transformación organizacional más eficientes y adecuados a la demanda, productos que cubren mejor las necesidades, herramientas más adecuadas para manejar fenómenos complejos y finalmente organizaciones más competitivas? Definitivamente sí, la complejidad de los problemas de las empresas se va acentuando con el tiempo, las herramientas tradicionales (lineales) para enfrentarlos ya son insuficientes y hasta contraproducentes.

Se requiere de un nuevo enfoque, de un nuevo concepto de ver al mundo y sus complicadas interrelaciones, de organizar mejor la fuerza productiva de las empresas y orientarlas hacia un nuevo objetivo común compartido por todos los involucrados, preocupándose además, de su desarrollo personal.

El objetivo principal no es sólo hacer más productiva o eficiente a la empresa. Eso ya no es suficiente. Ahora necesita ser competitiva, que los individuos que la integran sean capaces de aprender a aprender, de habituarse al cambio como un estado permanente de estilo de vida, que trabajen en equipo y en forma comprometida y motivada. Estos son los fundamentos para contar una empresa que tenga la capacidad de aprender, que no-solo se transforme organizacionalmente para reaccionar al medio ambiente sino que tenga la capacidad de incidir y provocar los cambios (empresa proactiva).

Las empresas deben ver más allá de la tradicional cadena cliente - proveedor. El pensamiento sistémico aporta la metodología para lograr algo más complicado que la simple satisfacción del cliente, su éxito.

Como se ve, son muchos los puntos que tienen que atacarse en forma paralela y en cada uno se debe cuidar la forma, el fondo y el tiempo (demora) para hacerlo. Por la complejidad que esto representa no se pude llevar a cabo sin una herramienta adecuada que nos permita ver las situaciones desde el punto de vista global, holístico. El Pensamiento sistémico es la herramienta fundamental para coordinar las actividades necesarias para lograr esta competitividad.

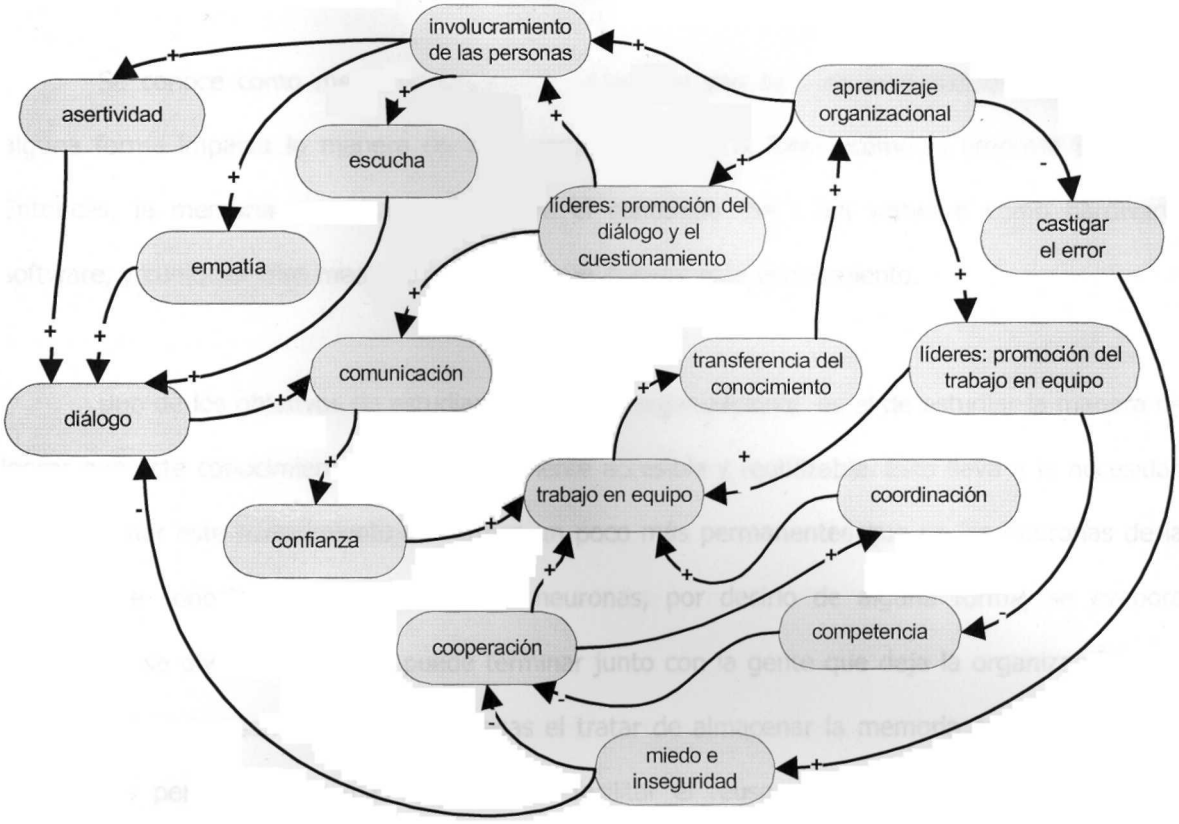


Figura 4.8. Mapa de influencias o arquetipo de una Organización Aprendiente. Este mapa solo muestra algunas de las variables interpersonales involucradas, tomando como punto de partida el papel de los líderes, como promotores de una cultura basada en el diálogo, el cuestionamiento y el trabajo en equipo.

7. Memoria Organizacional

Las organizaciones manejan muy diversos tipos de conocimiento para llevar a cabo sus funciones. Se encuentran, entre otros, conocimiento acerca de sus procesos, conocimiento acerca del diseño de sus productos y servicios y cómo mejorarlos, conocimiento relacionado con experiencias que tengan en la solución de problemas, o en la toma de decisiones, conocimiento acerca de su estrategia competitiva, etc.

Típicamente, este conocimiento está almacenado en cientos de lugares incompatibles; como son, documentos, bases de datos o programas. Una gran parte de este conocimiento no está almacenado ni en forma computarizada, ni en documentos escritos. Está guardado en las cabezas de las personas, lo que se conoce como: wetware.

Se conoce como memoria organizacional al conjunto de todo este conocimiento que de alguna forma impacta la manera como la empresa opera, la forma como la empresa funciona. Entonces, la memoria organizacional incluye el contenido tanto del wetware como hardware, software, y cualquier otro medio en que logremos guardar este conocimiento.

Uno de los objetivos de estudiar la memoria organizacional, es el de estudiar la manera de lograr que este conocimiento sea más fácilmente accesible y reutilizable. Esto lleva a la necesidad de almacenar este conocimiento en formas un poco más permanentes que en las neuronas de la gente. Este conocimiento almacenado en neuronas, por decirlo de alguna forma, se evapora fácilmente, se olvidan las cosas, puede terminar junto con la gente que deja la organización, etc. Entonces, es interesante para las empresas el tratar de almacenar la memoria organizacional en forma más permanente y más formal, para facilitar el reuso del conocimiento y evitar estar reinventando la rueda.

Marquardt afirma: "El conocimiento es el alimento fundamental para la organización aprendiente. Es el elemento que permite crecer a la organización. Los individuos pueden llegar o irse, pero el conocimiento de valor no puede perderse o la compañía muere por inanición"[21]. El mismo autor propone un modelo para la organización aprendiente constituido por cinco grandes subsistemas: la organización, las personas, el conocimiento, la tecnología y, como centro del modelo, el aprendizaje.

a. ¿Realmente se puede almacenar el conocimiento?

Hay varios aspectos importantes acerca de esta idea de almacenar el conocimiento de las personas:

---

[21] Marquardt , Michael J. Building the Learning Organization, McGraw-Hill ,1995, pp.74.

- El conocimiento es un fenómeno humano. Al tratarlo de "capturar" en un medio formal, necesariamente se tiene una pérdida, que puede o no ser significativa según el caso.
- ¿Quién determina qué conocimiento vale la pena almacenar?. Desde luego, es conveniente almacenar "el conocimiento más valioso" para la empresa, pero ¿Cómo puede determinarse el valor del conocimiento? Hay que recordar que el valor se lo atribuyen personas, y distintas personas pueden asignarle distintos valores.
- ¿Cómo se determina si el conocimiento almacenado es aún vigente, o si ya se puede borrar?
- ¿Se deben almacenar varios puntos de vista sobre el conocimiento?
- ¿Qué sucede con el contexto que lo hace válido en ciertas circunstancias solamente?

Para apreciar algunas de estas dificultades, supongamos que decidimos documentar proyectos, para ver qué los hace exitosos. Consideremos este comentario: ..."Cuando documentamos algún proyecto podemos ser muy detallistas, pero pocas veces hablamos de las personas que intervinieron, de sus características personales (historia) y los efectos positivos o negativos que determinada situación provocó en ellos, de qué humor andaban cuando se les presentó la propuesta del proyecto y que humor traía el que la presentó, o cuál era el contexto en que la empresa estaba en ese momento"[22].

b. La administración del conocimiento

La administración del conocimiento (knowledge management) es el proceso por el cual el aprendizaje y experiencia de los individuos de la organización puede ser accesado, reflejado, compartido y utilizado con el fin de nutrir el conocimiento individual y, por ende, el valor de la organización. No cabe duda que la administración del conocimiento en la organización aprendiente está tomando cada día más importancia. En un estudio reciente se muestra que el 42% de las

[22] Anónimo.

empresas de Fortune 1000 poseen un CKO (Chief Knowledge Officer) lo que significa que están pendientes e implementando una arquitectura de administración del conocimiento bajo la infraestructura colaborativa que ya poseen.

c. ¿Qué es en sí la administración del conocimiento?

Revisando la literatura existente podemos encontrar varias definiciones del término "knowledge management". Marquardt propone que el subsistema del conocimiento de su modelo se refiere a la administración del conocimiento adquirido o generado en la organización. Incluye los procesos de:

- **Adquisición.** Se refiere a la colección de datos existentes e información desde dentro o fuera de la organización.
- **Creación.** Involucra nuevo conocimiento que es creado dentro de la organización a través de la solución de problemas y la perspicacia.
- **Almacenamiento.** Consiste en la codificación y preservación del conocimiento valioso de la organización para su acceso fácil por cualquier miembro de la misma, en cualquier tiempo y desde cualquier parte.
- **Transferencia y utilización.** Incluye el movimiento mecánico, electrónico e interpersonal de la información y el conocimiento, ya sea sin o con intención, a través de la organización así como su aplicación y uso por los miembros de la misma.

Según Doculabs la administración del conocimiento consiste de varias facetas:

- Recolección de la información.
- Organización de la información.
- Distribución y diseminación.
- Colaboración en los resultados.
- Refinamiento de la información.

Si continuamos revisando definiciones en todas veremos más o menos las mismas actividades relacionadas con el proceso de administración del conocimiento.

d. Tecnologías para la memoria organizacional

Con la proliferación de redes en las corporaciones y la revolución de estándares en Internet se ha incrementado la cantidad de conocimiento disponible para las corporaciones hasta llegar al grado de convertirse en una sobrecarga de datos, información y conocimiento intangible que muchas veces es difícil de extraer para su uso. Esto ha provocado que surja la necesidad de administrar correctamente esa información, de encauzar debidamente su movimiento en la corporación e incluso mezclarla con aplicaciones que permitan a las personas ser más productivas. El conocimiento e información disponible antes sólo para ciertas personas privilegiadas se encuentra ahora a disposición de muchas personas en las grandes corporaciones.

Según Gardner el conocimiento dentro de la organización puede tomar varias formas, incluyendo las habilidades intangibles de un maestro mecánico o los inspectores de control de calidad. La información que es tangible y puede ser codificada en depósitos corporativos casi siempre viene de dos fuentes: **estructurada**, por ejemplo de una base de datos, **y no estructurada**, como el correo electrónico, faxes, correo de voz, presentaciones por computadora y otros documentos basados en archivos. La información estructurada fue inicialmente almacenada en formatos propietarios y el acceso a los datos estaba limitado a muy pocas personas. El resultado es una gran cantidad de información sin ninguna manera útil de ser extraída, consultada o recuperada.

e. Una solución factible: el uso de las Tecnologías de Información

Brian Quinn, autor de The Intelligent Organization, afirma que la tecnología es el ingrediente más importante para administrar el conocimiento organizacional. El entendimiento y

uso de las tecnologías requiere el estudio del arte y las ciencias del aprendizaje, el descubrimiento, las comunicaciones, las tecnologías de información y las ciencias computacionales.

Es indiscutible que las organizaciones que sepan como apalancar la tecnología para mejorar su capacidad de aprendizaje tendrán una ventaja competitiva en un futuro.

El subsistema tecnológico del modelo de Marquardt incluye redes tecnológicas integradas y de soporte, así como herramientas de información que permiten el acceso y el intercambio de información y aprendizaje. Incluye procesos técnicos, sistemas, y estructuras para colaboración, entrenamiento, coordinación y otras habilidades del conocimiento. Contempla también herramientas electrónicas y métodos avanzados para el aprendizaje, como son conferencias computarizadas, simulación, colaboración soportada por computadora, todas ellas herramientas para crear la "autopista del conocimiento".

En el mencionado subsistema se contemplan tres aspectos muy importantes para el uso de la tecnología en la administración del conocimiento:

- **Tecnología de Información**. Tecnologías basadas en computadora usadas para adquirir, codificar, procesar, almacenar, transferir y aplicar datos entre máquinas, personas y organizaciones.
- **Aprendizaje basado en computadora**. Se refiere a video, audio, y entrenamiento basado en multimedia para liberar y compartir el conocimiento y habilidades desde el lugar de trabajo.
- **Sistemas Electrónicos de Soporte al Desempeño**. Llamados en inglés EPSS, utilizan bases de datos (texto, imágenes, o audio) y bases de conocimiento para capturar, almacenar y distribuir información a través de la organización así como el ayudar a los trabajadores a alcanzar el mayor nivel de desempeño en el menor tiempo posible, con el mínimo de personal de soporte.

Es importante el mencionar el impacto que tienen las tecnologías de información sobre la organización. En el libro de "The Corporation of the 1990s", Michael Morton identifica seis impactos principales de la TI sobre el lugar de trabajo y el aprendizaje en el trabajo. Las Tecnologías de Información:

1. Cambian la manera en que se hace el trabajo.
2. Permiten la integración de las funciones del negocio.
3. Causan cambios en el clima competitivo.
4. Presentan nuevas oportunidades estratégicas.
5. Demandan cambios básicos.
6. Fuerzan la transformación del negocio.

Un comentario adicional al papel que juegan y jugarán las Tecnologías de Información en las organizaciones aprendientes nos lo presenta Michael Dertouzos, uno de los padres de Internet, en su muy interesante libro: "Qué Será", diciendo:

"Sean o no frecuentes los jefes de conocimiento cuando florezca el mercado de la información, no hay duda de que los jefes de sistemas y sus equipos de tecnología de la información seguirán presentes. El equipo de TI de mañana manejará una fracción menor del total de las actividades de información de la organización que su antecesor – el equipo de TI de hoy – porque habrá mucha gente de la organización que utilizará directamente el mercado de la información para realizar su trabajo"[23].

[23] Dertouzoz, Michael, Qué será, Planeta, 1997, pp. 128.

f. Trabajo Colaborativo Asistido por Computadora y el Groupware

Antes de iniciar este tópico, es conveniente el diferenciar y dejar claras algunas definiciones sobre el término groupware y el trabajo colaborativo asistido por computadora.

El trabajo colaborativo asistido por computadora (Computer-Supported Cooperative Work o CSCW) es el estudio relacionado con el trabajo de las personas usando la tecnología computacional. Algunos tipos representativos de aplicaciones incluyen el correo electrónico, sistemas de notificación y aviso, videoconferencia, sistemas de charlas (chat), juegos con multi-jugadores, y aplicaciones compartidas en tiempo real (como la escritura y dibujo colaborativos).

Casi siempre el CSCW se divide en dos dominios principales:

- **Trabajo sincrónico** (o en tiempo real), el cual considera a las personas que están trabajando juntas al mismo tiempo (por ejemplo con la videoconferencia).
- **Trabajo asíncrono**, el cual considera a la gente coordinando sus esfuerzos a través de largos periodos de tiempo (como el correo electrónico).

En ocasiones se utiliza el término CMCW (Computer Mediated Communication and Work) de manera indistinta para denotar al propio CSCW.

Por otro lado, el trabajo en grupo o groupware es el término utilizado a menudo para denotar la tecnología que las personas utilizan para trabajar juntos; el CSCW se refiere al campo que estudia el uso de dicha tecnología.

El groupware soporta los esfuerzos de equipos y otros paradigmas que requieren que las personas trabajen juntas, aunque en realidad no sea realmente "juntas" en el espacio y el tiempo. El groupware maximiza la interacción humana mientras que minimiza la interferencia de la tecnología.

El figura 4.9 nos muestra una vista desde ambas perspectivas: la humana y la de la tecnología:

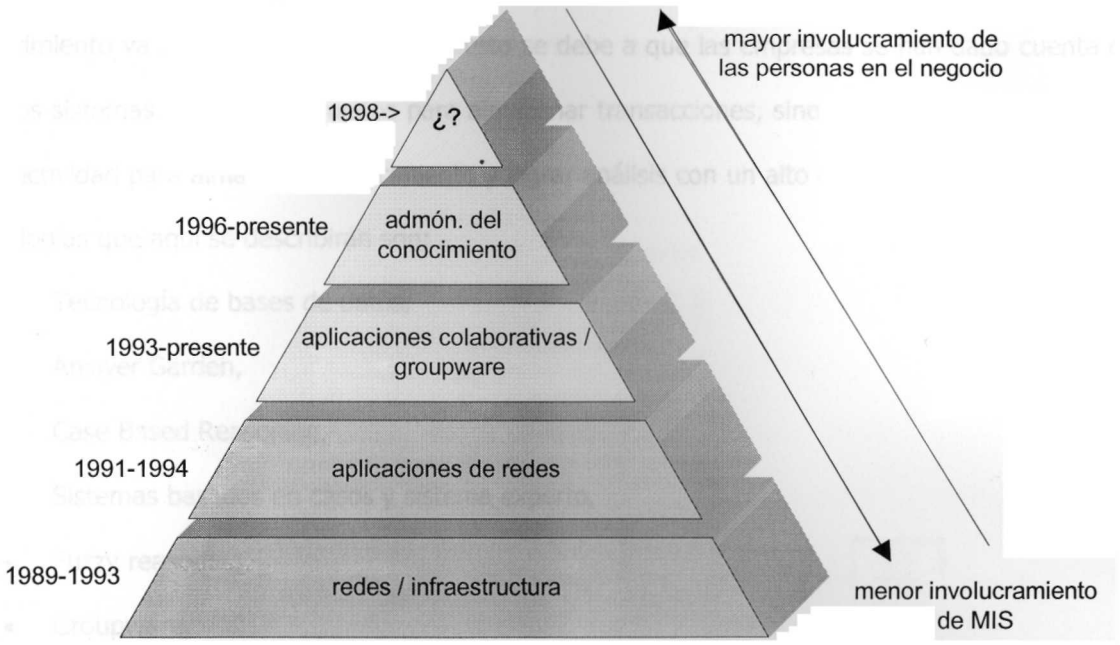


Figura 4.9. Diagrama evolutivo.

8. Relación con la Administración del Conocimiento

Dado que la administración del conocimiento es inherentemente una actividad de tipo colaborativo, y dada la oportunidad para la administración del conocimiento entre grupos de cuatro o más personas que puede ser realizada con tecnologías colaborativas de alguna forma, pueden ser tan simples como discusiones en correo electrónico o tan sofisticadas como bases de conocimiento de flujo de trabajo (workflow knowledge bases).

Vale la pena aclarar un punto importante: las tecnologías colaborativas por sí solas no facilitan la administración del conocimiento. El conocimiento debe ser relevante al trabajo y hacia las metas y objetivos de todo el negocio y debe ser accesible en las formas correctas y en el tiempo adecuado.

a. ¿Qué tecnologías de información pueden ayudarnos a llevar a cabo este almacenamiento, acceso, y reuso de la memoria organizacional?

La cantidad de tecnologías que se están utilizando para almacenar y rescatar el conocimiento va aumentando día con día, esto se debe a que las empresas se han dado cuenta de que los sistemas no solo son buenos para almacenar transacciones, sino también han comprobado su efectividad para almacenar conocimiento y lograr análisis con un alto grado de confiabilidad. Las tecnologías que aquí se describirán son:

- Tecnología de bases de datos,
- Answer Garden,
- Case Based Reasoning,
- Sistemas basados en casos y sistema experto,
- Fuzzy reasoning,
- Groupware,
- Intranets,
- Data Warehouse

b. Tecnología de Bases de Datos

Comencemos con la tecnología, probablemente más difundida**: la tecnología de bases de datos**. Un conocimiento que las empresas necesitan guardar, por ejemplo, es conocimiento acerca de dónde esta el conocimiento tácito, qué cabezas son las que lo tienen, quién es el que sabe qué cosa. Se puede utilizar un sistema manejador de base de datos para almacenar esa información. Supongamos que las personas llenan una forma, que incluye entre otros campos, una lista de palabras clave que nos indica los conocimientos e intereses particulares que esa persona tiene y que guardamos todos estos registros en la base de datos. Cuando se necesiten personas que conozcan de un determinado aspecto, o que sepan cómo resolver un cierto problema, se realiza una consulta a la base de datos y obtener la lista de registros correspondientes: tal vez el

lugar a donde se encuentran esas personas, tal vez, si están o no ocupados en otros proyectos, o si pueden ayudar, etc.; éstas son algunas de las aplicaciones de la tecnología de bases de datos.

c. Answer Garden

Un sistema exitoso de memoria organizacional, que utiliza la tecnología de bases de datos en dos diferentes formas, es el sistema que se llama **Answer Garden**, el "Jardín de las Respuestas", de Ackerman.

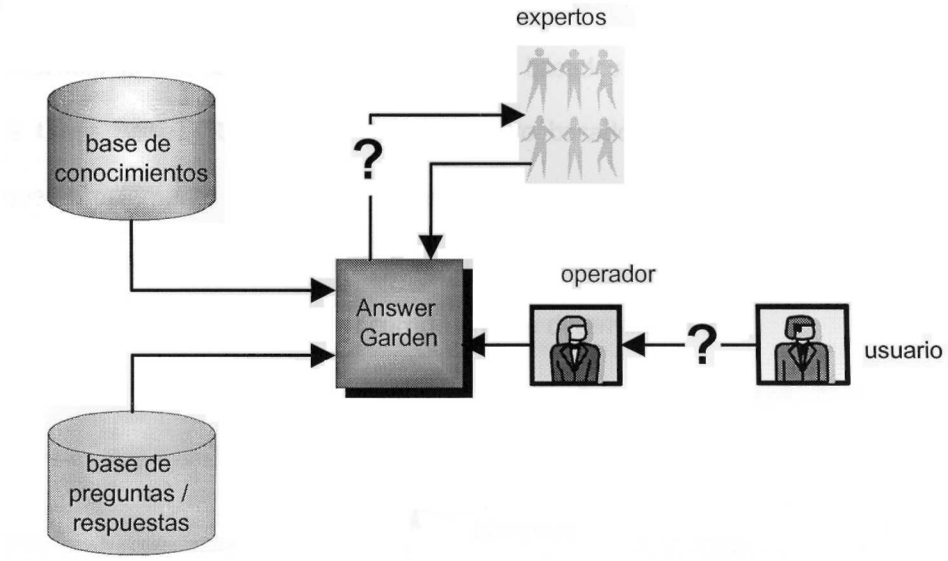


Figura 4.10. Answer Garden de Ackerman.

Ackerman utiliza dos bases de datos: una base de conocimientos o capacidades, que consideraremos que es la base de datos que nos indica el expertise, o que nos indica dónde está localizado el conocimiento de las diferentes personas; y una base de preguntas y respuestas. Un usuario, una persona que tiene alguna pregunta, se la haría a un operador que está manejando el Answer Garden, para ver si de esa pregunta ya se conoce la respuesta; en ese caso, el Answer Garden la recuperaría de la base de preguntas y respuestas. En caso de que esa pregunta todavía no esté registrada en la base de preguntas y respuestas, el Answer Garden está ligado, por correo electrónico, con todos los expertos de la compañía y con la base de conocimientos. Puede entonces

determinar quién puede saber la respuesta a esa pregunta, hacer la pregunta por correo electrónico al experto, y una vez que se tiene la respuesta, almacenarla en la base de preguntas y respuestas. De esta manera, se va acumulando, en la base de preguntas y respuestas, el conocimiento que los expertos tienen acerca de las preguntas; va creciendo esa memoria organizacional formal, y va creciendo a la medida de las necesidades de los usuarios que están buscando solución a sus problemas.

d. Case Based Reasoning

Otra tecnología que se utiliza es la de sistemas basados en casos, o en inteligencia artificial denominados **Case Based Reasoning** (Razonamiento basado en casos). Para situaciones más complejas que simplemente preguntas y respuestas, un sistema basado en casos permite almacenar una serie de situaciones organizacionales que pueden ser bastante complejas y que se conocen como los casos previos que han ocurrido en una determinada área

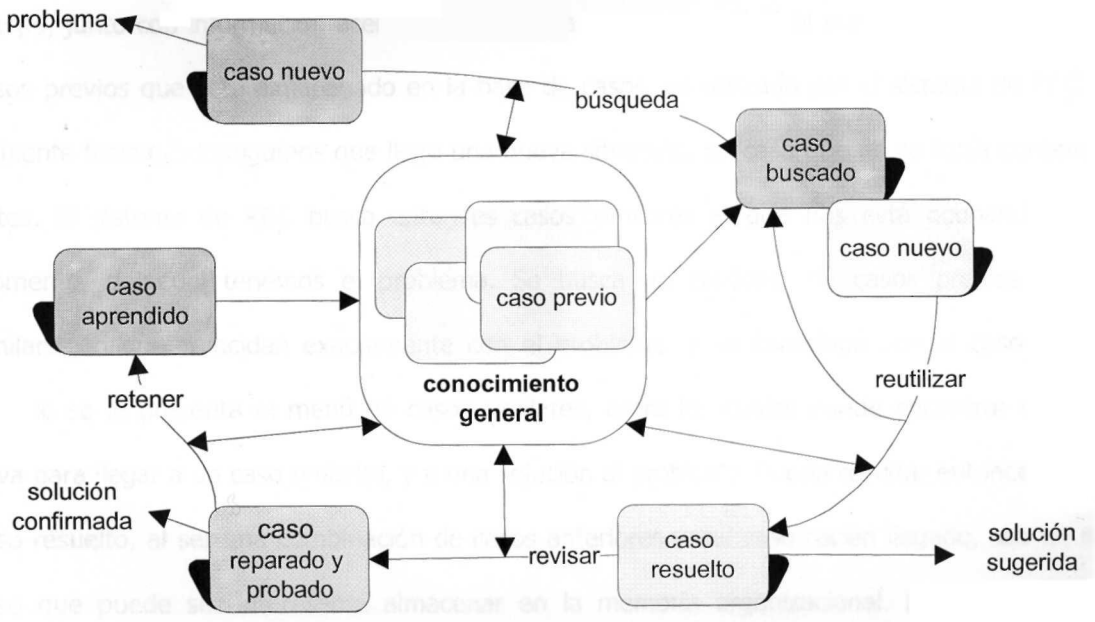


Figura 4.11. El ciclo de RBC.

La estructura de un caso varía según el tipo de conocimiento que se esté almacenando, pero una estructura típica genérica se compone de:

1. nombre del caso,
2. clasificadores,
3. antecedentes,
4. contexto,
5. problemática,
6. secuencia de eventos que ocurrieron para resolverla,
7. resultados,
8. otras situaciones donde podría aplicarse la misma solución,
9. lecciones aprendidas,
10. y personas o departamentos que pueden interesarse en el caso.

Por ejemplo, se pueden considerar situaciones de descompostura de un determinado equipo, junto con información acerca de la manera como se resolvió el problema. Este conjunto de casos previos que está almacenado en la base de casos, es utilizado por el sistema de RBC en la siguiente forma: Supongamos que llega una nueva situación, un caso que no se tenía contemplado antes. El sistema de RBC busca entonces casos similares al que nos está ocupando en este momento, del cual tenemos el problema. Se busca un conjunto de casos previos que sean similares, o que coincidan exactamente con el problema, y se combinan con el caso nuevo. Al usuario se le presenta el menú de casos similares, entre los cuales puede encontrar uno que le sirva para llegar a un caso resuelto, y a una solución al problema. Puede resultar entonces que este caso resuelto, al ser una combinación de casos anteriores y del caso recién llegado, sea un nuevo caso que puede ser interesante almacenar en la memoria organizacional. Entonces, este caso nuevo se aprende, es decir, se vuelve a guardar en la base de casos, que otra vez va creciendo conforme van ocurriendo estas situaciones.

Como vemos en la figura 4.11, un sistema de RBC requiere de un método para clasificar, indexar, almacenar y recuperar los casos; medidas de similitud entre unos casos y otros; y una forma de revisar casos existentes para mejorarlos y encontrar y retener casos nuevos.

e. Sistemas Basados en Casos y sistemas expertos

Los **sistemas basados en casos** son relativamente nuevos, comparados con otra forma de almacenar experiencia, que es la de los **sistemas expertos**, mal llamados expertos porque dan la idea de tener toda la expertise de un ser humano... esa era la intención, poder capturar toda la experiencia que tiene un humano y ponerla en una computadora. Aunque no se ha logrado ese objetivo, este tipo de sistemas sí ha tenido utilidad para almacenar conocimiento experto en dominios específicos. Por ejemplo, los sistemas expertos han tenido bastante éxito en aplicaciones de diagnóstico, entre otras. Imaginemos que tenemos en nuestra planta, que utiliza muchas calderas, un experto en reparar las calderas. Es una persona que ve alguna falla que tenga la caldera e inmediatamente ya sabe qué es lo que pasa y cómo hay que repararla. Supongamos que nos interesara capturar esa experiencia del experto en un sistema computacional, para no estar requiriendo de ese experto todo el tiempo, y poder reutilizar su conocimiento.

Entonces, este podría ser un muy pequeño fragmento de este sistema experto:

IF está echando humo THEN checar quemador

(...)

IF salida de aire tapada THEN destaparla

Los sistemas expertos están basados por lo general en un conjunto de reglas, que son proposiciones if-then básicamente: si sucede algo entonces debe ocurrir otra cosa o hágase esto. Una de las reglas de ese sistema de diagnóstico de calderas podría ser: si la caldera está echando humo, entonces algo se está quemando, vete a checar el quemador; y otra de las reglas podría decir: si la salida de aire está tapada, entonces destápala, y ¡ya se resolvió el problema del humo!.

Obviamente esta es una situación sumamente simple, que se describe únicamente para entender cómo podemos captar ese expertise de diagnóstico en forma de un conjunto de reglas if-then.

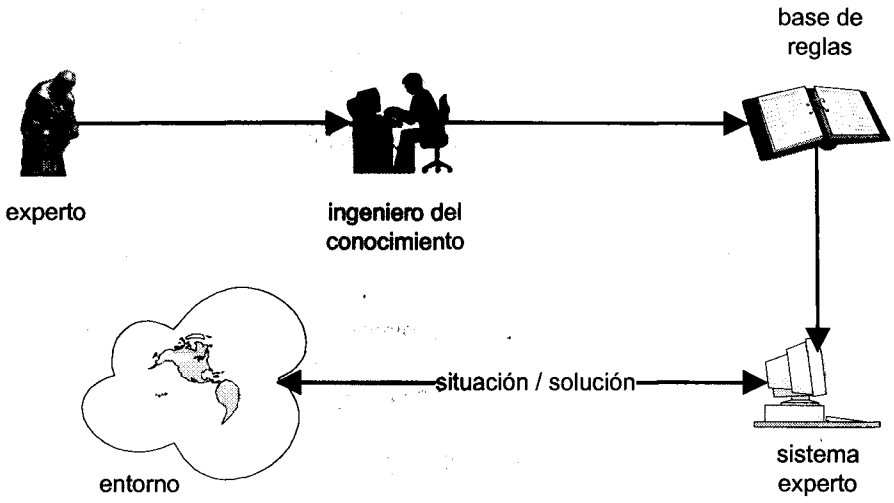


Figura 4.12. Proceso de extracción del conocimiento.

Algunas de las facilidades que tienen estos sistemas expertos, es la de dirigir ellos mismos el proceso de diagnóstico; por ejemplo, al llegar a la regla del quemador, el sistema mismo podría estar haciendo las preguntas al usuario, indicándole qué es lo que debe checar, a dónde debe checarlo, etc. Además, la consecuencia de una regla puede generar o afectar otras reglas, lo cual da a estos sistemas bastante poder. El proceso para llegar a tener este sistema experto, es más o menos el indicado en la figura 4.12.

El experto trabaja con un ingeniero del conocimiento, quien le hace preguntas, para determinar cuál es la expertise del experto, y entonces el ingeniero del conocimiento produce este conjunto de reglas, que son utilizadas por el sistema experto para que, cuando se presente una nueva situación, el sistema experto pueda dar una solución a través de un diálogo interactivo con el usuario.

Este proceso de extraer el conocimiento del experto ha resultado sumamente complejo. Los expertos en general actúan en forma instintiva; tienen todas esas reglas contenidas en su cerebro en tal forma que no tienen que pensar en ellas, y cuando el ingeniero del conocimiento le dice al experto: oye, piensa cómo estás pensando, pues al experto le cuesta mucho trabajo articular esa experiencia en forma de reglas. Los sistemas basados en casos, por otro lado, no tienen ese problema, porque todo lo que el experto tiene que hacer es describir los casos en una forma mucho menos estructurada que a través de una serie de reglas.

f. Fuzzy Reasoning

Una cosa que sucede con el conocimiento es que no es tan preciso como la tecnología computacional, es algo vago, a veces las cosas son así, y a veces son de una forma completamente diferente. Para atacar este problema de la vaguedad del conocimiento, se ha desarrollado un área que se denomina **fuzzy reasoning**. Fuzzy es un término que se podría traducir como difuso, vago o impreciso. Se ha utilizado este fuzzy reasoning o razonamiento fuzzy, en el área de inteligencia artificial combinado con varias tecnologías.

Tenemos por ejemplo fuzzy databases, bases de datos fuzzy, en las cuales un cierto registro no está determinantemente insertado a una cierta tabla, sino que pudiera estar en una o en otra con ciertas probabilidades: quizás una empleada que a veces trabaje de secretaria, a veces de administradora, o un estudiante que a veces sea maestro, pues no se puede asignar fácilmente ni a la categoría de maestro ni a la categoría de estudiante; entonces podemos decir pues está medio fuzzy, a veces es esto, a veces es lo otro. También tenemos fuzzy queries, consultas fuzzy, que nos permiten dar respuestas que no tienen la precisión de la respuesta de una base de datos, sino que nos pueden decir pues la respuesta a tu pregunta, con tal probabilidad es esta, pero puede ser con tal probabilidad esta otra. Y también tenemos fuzzy rules en los sistemas expertos, que nuevamente nos indiquen las probabilidades y situaciones con las cuales pueden o no ocurrir determinadas cosas.

g. Groupware

Ahora analicemos otra tecnología que es muy importante para la memoria organizacional, que es la tecnología del **groupware**. Jeff Conklin dice que las empresas son muy buenas para almacenar todo tipo de documentos finales, pero no para almacenar las decisiones y las suposiciones relevantes para generar estos documentos. Consideren por ejemplo una empresa que se dedique a diseñar por ejemplo, software, aviones, etc. En ese proceso de diseño, se toman múltiples decisiones. Lo que la empresa típicamente almacena son los resultados del diseño en forma de varios documentos, planos, etc., pero no se almacena toda la cantidad de decisiones que se tuvieron que tomar ni por qué se tomaron esas decisiones. Y no se almacena, primero porque todo esto es el expertise de diseño precisamente de la empresa; está en la cabeza de la gente y no tenemos forma de almacenarlo, y porque el proceso de diseño se ejecuta entre seres humanos, hablando, platicando, y las palabras se las lleva el viento, no quedan almacenadas en ningún lado.

Conklin propone que los ingenieros de una empresa de diseño, o las personas de una organización, se comuniquen entre ellos, no tanto en forma presencial, sino a través de sistemas de groupware. Estos sistemas de groupware son sistemas computacionales que permiten que grupos de personas interactúen unas con otras.

Por ejemplo, vamos a suponer que se reúne un grupo de personas en esta empresa de diseño para tratar de decidir de qué material vamos a hacer la caldera. Una posibilidad sería que se pusieran a platicar, pero otra posibilidad sería que utilizaran un sistema que les estructure en alguna forma la discusión que van a tener; por ejemplo, alguien podría anticipar que las posibles respuestas al material del cual se va a hacer la caldera podrían ser de latón o de acero, o de algún otro material, que a alguien se le ocurra. Alguien podría anticipar que una vez que se esté discutiendo determinado material, las respuestas se van a orientar hacia ver las ventajas y desventajas de ese material. En otras palabras, antes de reunirnos a platicar, estamos elaborando

lo que podríamos llamar una estructura de discurso de cómo va a estar la discusión. Si esa discusión la hacemos entonces en forma electrónica, ya sea que todos estemos en el mismo cuarto, o estemos separados en diferentes partes de la oficina, pero la hacemos en forma electrónica, esta estructura nos puede indicar hacia dónde vamos a orientar nuestros comentarios, siempre dejando una oportunidad de poner otros comentarios y de poner otros materiales, inclusive de discutir alguna otra cosa. Sin embargo, observen que una vez que esta estructura está llena, ya con todas las opiniones de todos los miembros, hemos guardado el conocimiento de diseño, hemos guardado las razones por las cuales se tomaron determinadas decisiones de diseño, y ese conocimiento estaría disponible para ser accesado después, un conocimiento que antes se hubiera quedado almacenado en las neuronas ahora lo tenemos almacenado en nuestro sistema de groupware.

Una Taxonomía para el Groupware

Según la revista Collaborative Strategies de San Francisco Cal; el groupware se puede clasificar en 12 categorías funcionales de acuerdo a servicios de groupware, aplicaciones de groupware y aplicaciones y productos colaborativos emergentes basados en Internet.

1. **Correo y mensajería electrónica.** Incluye infraestructuras para mensajería y sistemas de correo electrónico.

cc:Mail -- Lotus	Eudora – Qualcom
Microsoft Mail/Exchange	QuickMail – CE Software
Banyan Intelligent Mail – Banyan	OracleMail – Oracle

2. **Calendarización de grupos.** Productos para calendarios, reuniones, juntas y coordinación de recursos.

Lotus Organizer-IBM/Lotus	OnTime-FTP Software
Synchronize-CrossWind Technology	Microsoft Schedule + Network
Meeting Maker-On Technologies	Scheduler-CE Software

Pencil Me In-Sarrus Software	CaLANdar-Microsystems Software

3. **Sistemas para reuniones electrónicas. (Electronic Meeting Systems (EMS)).** Sistemas de conferencia en tiempo real (local y remota) así como sistemas de presentación colaborativa.

Group Systems-Ventana	MeetingWorks 2-Enterprise Solutions
Council Services-	CoVisionOption Finder-Option Technologies
Facilitate.com-McCall Szerdy Assoc.	TeamTalk-Trax SoftWorks

4. **Conferencia de datos en tiempo real y en escritorio.** El enfoque es sobre el tiempo real, más que un BBS o Notes. Estos productos almacenan documentos, y/o permiten a otros el ver y trabajar simultáneamente sobre los documentos, ya sea en las pantallas de los otros usuarios o sobre un pizarrón electrónico.

ShowMe-Sun Solutions	RoundTable-ForeFront Group
Aspects-Group Logic, Inc.	Being There-Intelligence at Large
NetMeeting-	MicrosoftPictureTalk-Picture Talk
CoolTalk-Netscape	FarSight-Databeam

5. **Conferencia en tiempo no real.** En este tipo de productos se lleva a cabo una conversación a través del tiempo, dejando un mensaje para alguien y ese alguien lo contesta, se puede contestar a esa persona más tarde. Los mensajes pueden ser públicos (como en un BBS) o privados (como en una base de discusión de Notes).

TeamTalk-Trax SoftWorks	WebBoard-O'Reilly
Pacer Forum-NetManage	WebShare-RadNet
Lotus Notes-IBM/Lotus	FirstClass-SoftArc Inc.

InterOffice-Oracle	News Server-Netscape

6. **Manejo de documentos de equipos.** Edición en grupo, trabajo de edición con pantalla compartida, administración de documentos e imágenes en grupos y de bases de datos de documentos.

Face-to-Face-Crosswise	Documentum-Documentum, Inc.
MarkUp-Mainstay Software	OnGo Document Management-Uniplex

7. **Workflow.** Diagramación de procesos de workflow, herramientas para análisis, productos para enrutamiento de formas electrónicas.

Workflow Analyst-Action	Technology JetForm-JetForm Corp.
Staffware for Windows-Staffware	Formflow-Symantec
Open Workflow-Wang	Metro-Action Technologies
Workflow BPR-Holosofx	Flowmark-IBM

8. **Herramientas para desarrollo y utilerías para Groupware.** Utilerías para soportar el trabajo en grupo, acceso remoto a la computadora de alguien más del grupo y herramientas específicas para el desarrollo de aplicaciones de groupware.

Windows for Workgroup-Microsoft	CoEX-Twin Sun
Lotus Notes-Lotus Replication	Reporter-Ernst & Young
InterOffice-Oracle	ReplicAction-Cassal

9. **Marcos de referencia para Groupware.** Estos productos ayudan a integrar "islas de colaboración" para poder implementarlas a través de varias plataformas computacionales, sistemas operativos, y diferentes arquitecturas de redes y de correo electrónico.

GroupWise-Novell Lotus	Notes-Lotus/IBM
TeamOffice-ICL/Fujitsu	OpenDoc-Apple/IBM
GoldMetal Workgroup-Decathlon	OpenMind-Attachmate

10. **Servicios de Groupware.** Servicios para soportar la colaboración.

Planning and Implementatio	Business Process Re-Engineering
Application Development	Knowledge Management
Training and Maintenance	Electronic Meeting Facilitation
Change Management Consulting	Consulting

11. **Aplicaciones de Groupware.** Aplicaciones verticales que usan tecnologías colaborativas ya sea para mejorar procesos o soportar la colaboración en un ambiente específico de trabajo.

BAI-5000 Distribution Manager	System-Bussiness Autoring
Reposity Technologies CenterPoint	Patient Tracking System
Directions ProTEAM-Scopus	CustomerFirst

12. **Aplicaciones colaborativas basadas en Internet.**

InterNotes Publisher-IBM/Lotus	PCS 50-PictureTel
RoundTable-The ForeFront	Group Metro-Action Technologies
SamePage-WebFlow	

Algunas compañías y algunos productores de software ven al groupware simplemente como el compartir información a través de compartir bases de datos como núcleo central del groupware. Una arquitectura simple que puede servir como un inicio al groupware es el sistema de

administración de información distribuida, aunque el concepto de trabajo en grupo va más allá. Una arquitectura típica de los sistemas mencionados es la siguiente:

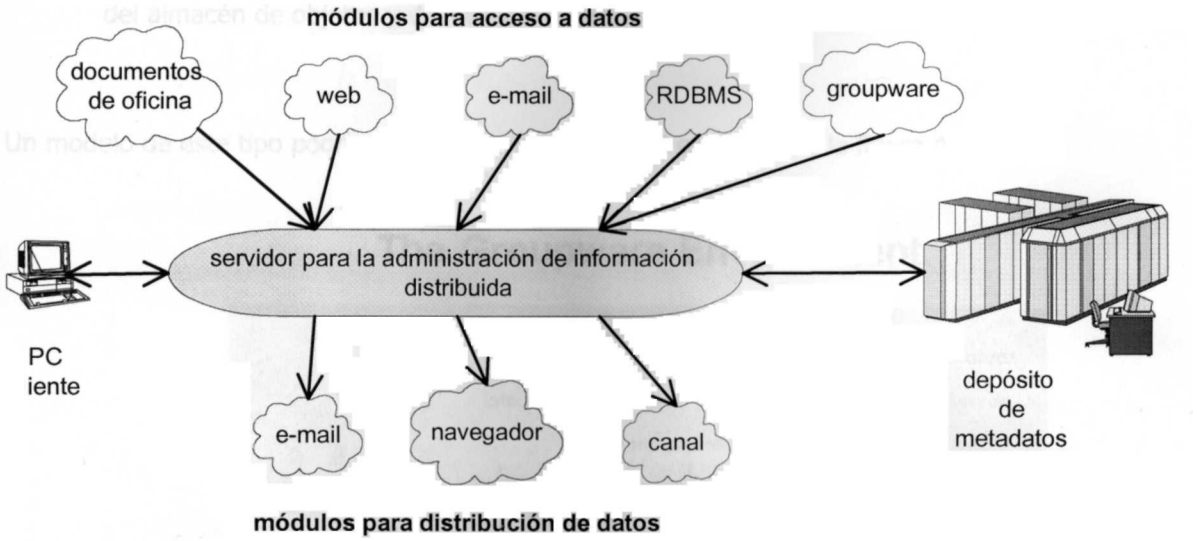


Figura 4.13. Sistema para la administración de información distribuida (DIMS), fuente: Plumtree Software.

Una arquitectura de este tipo permite el compartir datos, pero no cumple con los fundamentos del groupware, el cual es una integración de varias tecnologías. Un marco de referencia general para el trabajo en grupo debe incluir:

- **Comunicación:** Alta capacidad de mensajes electrónicos.
- **Colaboración:** Facilitar un espacio de trabajo virtual rico y compartido.
- **Coordinación:** Añadir la estructura de procesos de negocio a la comunicación y colaboración, así como también implementar políticas de la compañía.

De ahí que una plataforma para trabajo en grupo esté representada por tres tecnologías primarias:

- Un **almacén de objetos** en el cual el conocimiento corporativo –mensajes, documentos, formas, memorándums, reportes—puedan almacenarse y administrarse.

- Un **modelo de acceso y distribución** que permita a los usuarios localizar fácilmente la información así como diseminarla.
- Un **marco de desarrollo de aplicaciones** que apalanque los servicios nativos existentes del almacén de objetos y del modelo de acceso/distribución.

Un modelo de este tipo podría estar representado por el esquema de la figura 4.14:

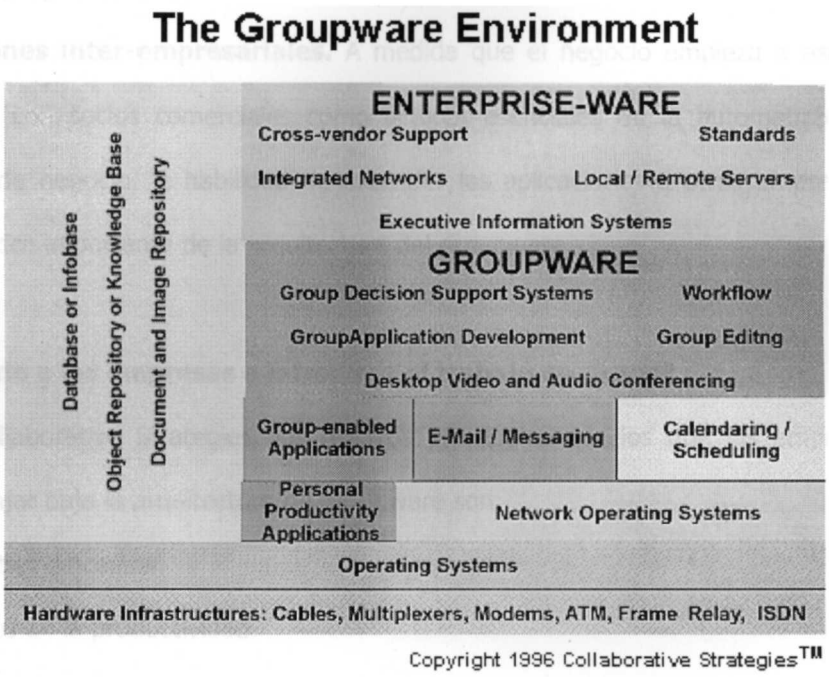


Figura 4.14. El medio ambiente del groupware.

De aquí que existan algunos requerimientos generales para los ambientes de trabajo en grupo (según Lotus Development Corporation[24], 1998):

- **Integración con recursos externos.** El punto de origen para la información de grupos es a menudo una fuente externa al ambiente propio del groupware (por ejemplo:

[24] Lotus Development Corporation, Información de productos Lotus. URL:http://www.lotus.com/ 30.04.1998.

herramientas de productividad personal en computadoras de escritorio, bases de datos relacionales, etc.).

- **Independencia de la plataforma.** Mientras que las aplicaciones de groupware inician casi siempre como implementaciones departamentales, muchas de ellas resultan eventualmente en un despliegue a toda la compañía.
- **Movilidad.** Una infraestructura de groupware debe ser capaz de soportar varios sitios dispersos geográficamente, incluyendo hogares y computadoras portátiles.
- **Aplicaciones inter-empresariales.** A medida que el negocio empieza a asociarse con clientes y con socios comerciales como actores esenciales en la automatización de los procesos de negocio, la habilidad de extender las aplicaciones a otras empresas es una característica importante de la arquitectura del groupware.

¿Qué ha motivado a las empresas a introducir el trabajo en grupo?

Según Collaborative Strategies, los motivos principales por los que las empresas están cambiando a trabajar bajo la arquitectura de groupware son:

- Mejor control de costos.
- Incremento en la productividad.
- Mejor servicio al cliente.
- Soporte para la administración de la calidad.
- Menos juntas.
- Automatización de procesos de rutina.
- Extender la corporación para incluir a clientes y proveedores.
- Integración de equipos dispersos geográficamente.
- Incrementar la productividad a través de reducir el tiempo de entrega.
- Mejor coordinación global.
- Proveer un nuevo servicio que diferencie a la organización.
- Apalancar la experiencia profesional.

Y a todas ellas podemos agregar el crear un ambiente de aprendizaje dentro de la empresa por medio de la administración correcta del conocimiento generado. Estos mismos consultores nos proponen 20 reglas generales para implementar con éxito el groupware dentro de la empresa (se describen literalmente en inglés para no cambiar su significado):

1. Find a groupware champion! The higher in hierarchy, the better. Get management's hands on the keyboard. By getting top management involved, they see the benefits, and you get a lot more support!
2. Groupware changes the corporate culture. Plan for it!
3. Pick a pilot project rather than trying to roll groupware out to the whole organization.
4. Pick a bounded project with a group that is supportive of both technology and innovation.
5. Pick a project with visibility and financial impact
6. Realize that training, maintenance, and support will be the majority of the cost, rather than the initial cost of the software.
7. Measure productivity factors before and after the project has started. This is a good way to cost-justify groupware!
8. Pick groupware software based on a specific business problem that needs to be solved and has not been solved successfully using traditional methods. Corollary: Don't pick the groupware first and then find a problem.
9. Make sure you have adequate planning, support, training, and maintenance for your project.
10. No single groupware product can do it all. Don't expect it to!
11. Don't expect software vendors to offer you all the services you need for groupware. You may need to use internal people or consultants to ensure your project's success.
12. Groupware is not a quick fix! As part of a re-engineering effort, it may take 2-4 years to see the results.
13. Listen to the people involved in the pilot project. They are experts on what needs to be done and can often suggest ways to better the process.

14. Don't be afraid to make changes! A pilot project is an experiment. Learn as you go.
15. Make sure the software you pick fits with existing systems. Try to amortize your LAN investment by connecting to your mainframe or other legacy systems.
16. You can't change people overnight. Be prepared for resistance!
17. People take time to change. Organizations take even longer!
18. It takes courage to change a corporate culture! Applaud those who are willing to change.
19. Be careful about paving the cow path. There is no point in automating a very inefficient process. There are no big productivity wins here!
20. Groupware can be very political. Make sure it is a big win![25]

Se ha hablado hasta el momento de las ventajas que presenta el groupware, pero también se debe lidiar con una serie de fuerzas que inhiben el crecimiento de esta arquitectura dentro de las empresas, incluyendo:

- Bajo nivel de educación en la comunidad de negocios acerca del groupware.
- Confusión en el mercado con relación a la naturaleza del groupware. Mucha de la información conflictiva y competitiva distribuida por las compañías que venden groupware ha incrementado esta confusión.
- La recesión económica decremento los presupuestos y muchas compañías perciben que no pueden afrontar el presupuesto de gasto en groupware.
- Los canales de distribución para el groupware son nuevos y no están totalmente implementados.
- Los departamentos de MIS de las corporaciones tienen miedo de llegar a depender de un vendedor de groupware.
- Las organizaciones se resisten al cambio.

---

[25] Zack, Michael; Serino, M. Knowledge Management and Collaboration Technologies, Lotus Corporation, URL: http://www.lotus.com/services/institute.nfs/550137bfe37d25a18525653a005e8462/ 30.04.1998.

h. Intranets

Existen muy pocos estándares en el mercado del groupware para permitir un crecimiento rápido. Ahora consideraremos a las **intranets**. En este momento las empresas están implantando a toda velocidad intranets, que es básicamente una red internet dentro de la empresa. Vamos a hacer un pequeño paréntesis para considerar que esto de la memoria organizacional nos trae a la mente términos antropomórficos, al ver a la empresa como un organismo que tiene su memoria, que aprende, que es capaz de procesar información, de tomar decisiones. Si vemos a toda la empresa en conjunto en esos términos, las neuronas de la empresa serían los trabajadores. Tal como nuestro conocimiento no está, dicen los científicos, en las neuronas en sí, sino en gran medida en las interconexiones entre las neuronas, podemos también visualizar que en una empresa, una parte muy importante del conocimiento y potencial no está tanto dentro de cada cabeza, sino en las relaciones formales e informales tan diversas que establecen los individuos en una organización. Gran parte de la theory-in-use de la empresa, la forma como realmente ocurren las cosas, se encuentra en esas relaciones: esta persona trabaja con esta, es amigo de la otra, le pasa información formal a otro, le debe favores a alguien de otro departamento, recibe tips de uno ya retirado o promovido. Vamos a suponer que cada una de estas personas dentro de la organización tuviera una home page, una página en su internet, y que las conexiones, toda esa enorme variedad de interrelaciones que tienen las personas empresariales, estuvieran representadas por ligas de hipertexto asociadas con una identificación del tipo de relación. Tendríamos ahí, almacenada ya, en forma natural, una parte muy importante de la memoria organizacional, que consiste en ese conocimiento sobre las diferentes relaciones que guardan los individuos de la organización.

Hoy día muchas organizaciones utilizan una LAN para conectar computadoras personales a recursos compartidos, como servidores de archivos e impresoras de red. Una intranet extiende los recursos que se pueden compartir en una organización para incluir documentos, bases de datos, imágenes, videos, sonido y multimedia. Y se pueden compartir de forma segura entre su LAN y el

mundo utilizando Internet. Además, con la disponibilidad de interfaces de visualización, la posibilidad de tener una interfaz gráfica común para la creación, gestión y acceso a la informació se convierte en realidad. El tiempo de aprendizaje se reduce en gran medida y se pueden distribu aplicaciones instantáneamente por toda la organización tan pronto estén disponibles.

Una intranet permite nuevos mecanismos de toma de decisiones utilizando la información que se intercambia electrónicamente, en lugar de en papel, por teléfono o mediante reuniones en persona. El uso de información electrónica permite un acceso fácil y completo a la misma, así como a foros donde se participa intercambiando preguntas y respuestas, dentro de la organización. Las capacidades de cooperación ayudan a promover una interacción más efectiva entre grupos. Otras capacidades permiten presentar información multimedia y ahorrar tiempo de búsqueda en la recuperación de información.

Compartir información en una intranet es una excelente forma de que más empleados tengan responsabilidad directa en el proceso de comunicación dentro del departamento y de la compañía. La gente se va entusiasmando con este método cuando ven, no sólo que pueden ayudar a sus compañeros a realizar su trabajo mejor y más rápido, sino que aumenta también su propia productividad.

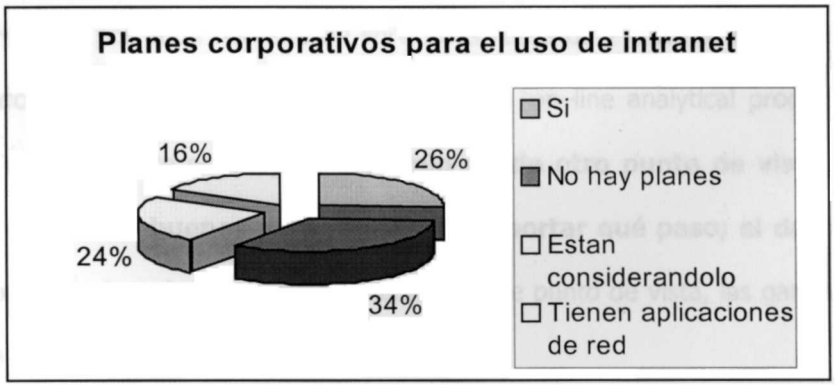
Con el paradigma de intranet se están desarrollando nuevas generaciones de herramientas que enlacen con las bases de datos corporativas preexistentes y proporcionen acceso asistido a información en distintos formatos (archivos de tratamiento de texto, gráficos, bases de datos, HTML, presentaciones) en cualquier lugar del entorno informático corporativo, desde computadoras personales a servidores de archivos corporativos y grandes computadoras.

De acuerdo con lo que menciona cio.com, los beneficios de un ambiente intranet son los siguientes:

Antes	Después
múltiples plataformas	independencia de la plataforma
múltiples formatos de datos	múltiples formatos de datos
múltiples interfaces	una interfaz
múltiples protocolos	protocolos comunes
Islas de información	fácil acceso a la información
difícil compartir información	fácil de publicar en toda la Intranet
difícil encontrar información	ambiente de mucha información
capacidades de plataforma diferentes	amplio rango/tipos de información
	acceso más fácil y rápido
	fácil entrenamiento, actualización

¿Cómo están reaccionando las organizaciones conforme a la utilización de una Intranet? Los administradores de los diferentes departamentos en una compañía han identificado rápidamente las ventajas de estos nuevos medios de comunicación como un recurso importante dentro de cualquier corporación. Forrester Research entrevisto de acuerdo con Fortune a 500 compañías y encontró que las dos terceras partes tienen todavía o están considerando el utilizar aplicaciones intranet. Estas compañías han identificado a intranet como un poderoso mecanismo para hacer información más valiosa a la compañía, esto se puede ver en la siguiente figura.

Figura 4.15. Planes corporativos para el uso de intranet, fuente: Forrester Research. 1996.



La intranet no es sólo un medio poderoso de comunicación, también es una base de conocimientos. Tiene la ventaja de que con una intranet es más fácil capturar y manejar conocimiento implícito y no estructurado, en contraste, las DBMS requieren de esquemas muy bien estructurados para ser efectivos.

Las formas en que aprendemos nos ayudan a entender cuales son los roles, habilidades, herramientas y procesos que necesitamos desarrollar para ayudar a los individuos en la organización a encontrar el conocimiento que todavía es valioso, que mueve a la organización hacia el aprendizaje y captura las experiencias en la base de conocimiento organizacional con el mínimo de esfuerzo.

i. Data warehouse

Para finalizar, se describirá el concepto de **data warehouse.** El data warehouse representa el último gran paradigma de la administración de bases de datos. Los primeros sistemas administradores de datos fueron jerárquicos, corrían en mainframes y su uso principal era el almacenamiento. El primer gran cambio vendría a inicios de los 80's con la adopción de sistemas de bases de datos relacionales en los cuales se desarrollaron aplicaciones operacionales primarias; estos sistemas corren típicamente en minicomputadoras y son utilizados para el procesamiento de transacciones en línea (on-line transaction processing, O.L.P.T.) como por ejemplo, pedidos de clientes, surtimiento y cobro. Ahora viene el data warehouse, su ejecución es del tipo cliente/servidor y por ende necesita de servidores robustos y una red de clientes, estos sistemas son utilizados para procesamiento analítico en línea (on–line analytical processing, O.L.A.P.) una aplicación esencialmente estratégica. **Viéndolo desde otro punto de vista, los sistemas de bases de datos son buenos para registrar y reportar qué paso; el data warehouse, nos dice el por qué.** Cuando los gerentes entiendan este punto de vista, las ganancias pueden crecer de manera interesante.

De hecho, un estudio realizado por International Data Corporation a las 45 mayores compañías, encontró que "el retorno sobre inversiones en data warehousing en un periodo de tres años era del 401%, este indicador es muy impresionante, pero al analizar las muestras encontramos valores que van desde un 16,000% cayendo hasta un -1,857%"[26].

Aunque International Data ha dicho que los resultados negativos pueden explicar debido a la adquisición de sistemas muy caros y poco usuales; su bajo nivel de uso o el requerimiento en tiempo de nuevos proyectos que muchas veces son grandes y complejos; además debemos recordar que la forma en la que se administra un proyecto nos puede decir mucho de su éxito o fracaso.

Por la parte cultural, el empleo de data warehouse empuja a los gerentes, directores, etc.; a nuevos roles y casi siempre fuerzan cambios en las relaciones entre vendedores y compradores. Lo anterior depende fuertemente del propósito con el cual se hizo el sistema. El data warehouse pudo haberse creado para mejorar la efectividad de un proceso, para cambiarlo o para crear uno nuevo. En muchos de estos casos los métodos tradicionales de control, como el retorno sobre la inversión, carecen de sentido.

Para ejemplificar, tomemos la información proporcionada por MCI, ellos llevan un registro de cada uno de sus clientes, en donde almacenan datos del cliente, servicios contratados, llamadas, destino, duración, etc. Esto se traduce en un sistema que almacena y maneja cerca de 3 terabytes de datos, lo suficiente para algo mas de 100 millones de registros. Ahora imaginemos el tiempo que se requeriría para tener los resultados de una consulta analítica compleja bajo el esquema de bases

---

[26] Fisher, Lawrence. Along the Infobahn. URL: http://www.strategy-business.com/ 04.07.1998.

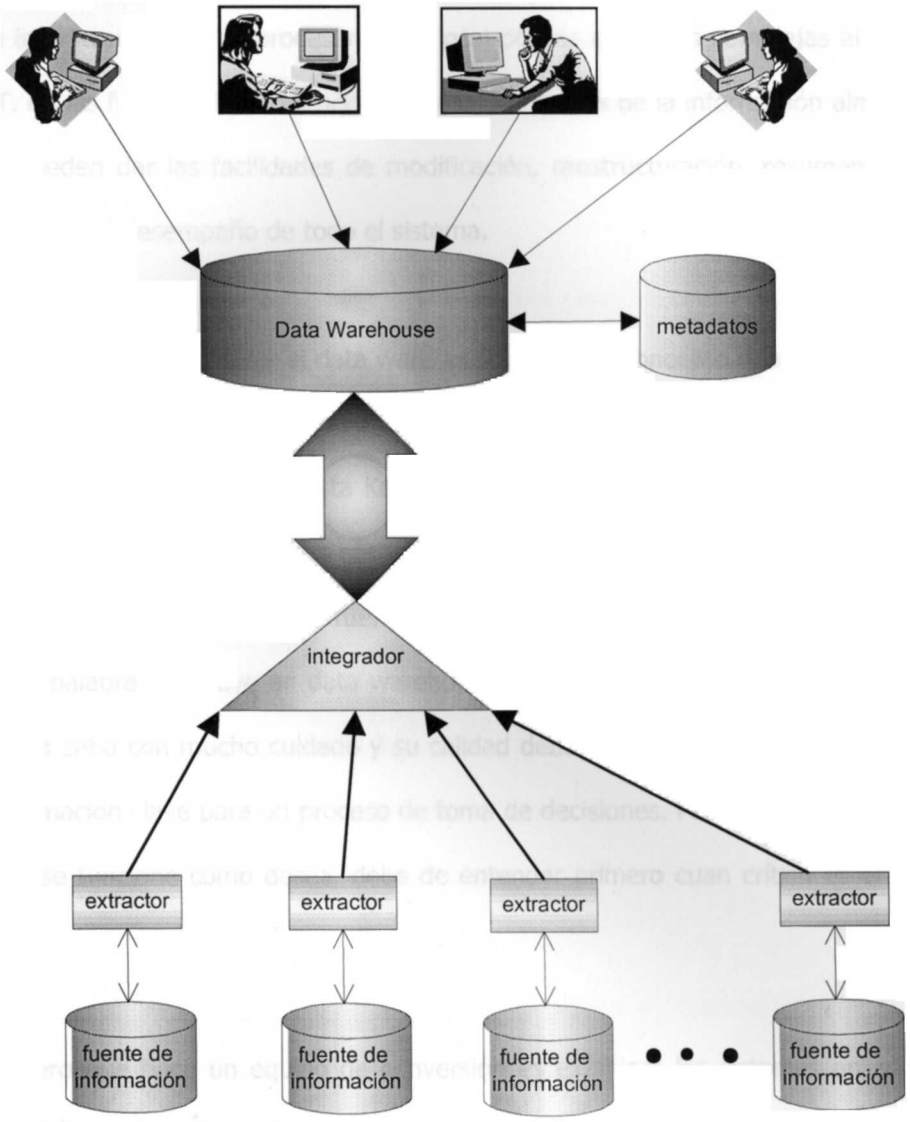
de datos relacionales ¡puede tardarse días!. Esto se debe a que las bases de datos fueron diseñadas primero, para procesar transacciones y como segundo para el procesamiento limitado de consultas a través del lenguaje SQL y, como su nombre implica, éste es muy efectivo si la consulta se apega a cierta estructura.

Pero las consultas analíticas son definidas vagamente; de hecho la primera pregunta que hace un analista, rara vez obtiene la respuesta correcta. Procesar una oración de consultas ad-hoc, sin estructura, hace que una base de datos convencional realice varios recorridos en todos sus registros, logrando con ello, un proceso consumidor de tiempo. Para mejorar el desempeño en tales aplicaciones los data warehouses estructuran los datos, en lugar de la consulta, con varios esquemas de índices, con ello el sistema puede responder rápidamente a consultas inesperadas.

Los data warehouses pueden ser optimizados para tareas analíticas precisamente porque no se utilizan para procesar transacciones y no necesitan mantener la consistencia (en cualquier punto del tiempo) que los sistemas de tipo O.L.T.P. deben tener; como el warehouse esta separado del sistema O.L.T.P. las consultas y la seguridad no se ve afectada ¿la razón?, el data warehouse es un concepto que solo extrae información, no la modifica.

Resumiendo, el data warehouse es un concepto, no es un producto que pueda comprarse en cualquier tienda de software. Este es un conjunto de software y hardware que pueden ser utilizados para analizar cantidades masivas de datos que las compañías acumulan para tomar mejores decisiones para el negocio.

Figura 4.16. Arquitectura del Data Warehouse.



Los data warehouses reúnen copias de información de varias fuentes remotas como son, datos distribuidos, autónomos, posiblemente heterogéneos, etc.; en una sola base de datos para consultas y análisis (ejemplo soporte de decisiones). Y el data warehousing es la técnica emergente para obtener e integrar los datos para construir el data warehouse: ya que en el enfoque tradicional de consultas, tiene muchas desventajas, como son, los retardos en el procesamiento de

consultas debido a las lentas y a veces inexistentes fuentes de información, la filtración compleja y su integración, por su diseño ineficiente y potencialmente caro para consultas frecuentes y procesamiento local; el data warehouse nos da la ventaja de un alto desempeño en las consultas, reduciendo la interferencia con el procesamiento local por las consultas complejas al warehouse y con el O.L.P.T. en las fuentes de información, además las copias de la información almacenadas en el warehouse pueden dar las facilidades de modificación, reestructuración, resumen, etc., de las consultas sin afectar el desempeño de todo el sistema.

Aunque se podría considerar al data warehouse como un concepto relativamente nuevo, ya existen fuentes de información acerca de los procesos de cambio y depuramiento de datos. Por ejemplo, en la revista DBMS la columnista Kathy Bohn hace un análisis de la conversión de datos para el warehouse y no dice entre otras cosas que éste proceso es muy complejo, consumidor de tiempo y nada agradable; pero que es la fuente de un buen sistema warehouse, ya que después de todo "dato es la palabra operativa en data warehouse"[27]. Las operaciones de conversión de datos se deben llevar a cabo con mucho cuidado y su calidad debe ser indiscutible ya que el warehouse contiene la información clave para un proceso de toma de decisiones. Para que una compañía logre que su warehouse funcione como desea, debe de entender primero cuan crítico es el proceso de conversión.

Lo primero que hace un equipo de conversión es examinar los sistemas anteriores y sus datos para de ahí pasarlos al warehouse; por ello los equipos de conversión sufren de presiones durante todo el proceso de implantación y después de todo su actividad es cuestionada por los resultados arrojados y como bien apunta Bohn "Typically, the conversion system is questioned because, as it's the newest system, users assume that the conversion system is what introduced the problems"[28].

---

[27] Bohn, Kathy, "Converting Data for Warehouses", URL: http://www.dbmsmag.com/9706d15.htm, 06.08.1998.
[28] Ob cit.

j. Aseguramiento de la calidad de los datos

El aseguramiento de la calidad de la información no puede ser visto como una actividad aislada durante el proceso de conversión de datos. El plan de conversiones debe especificar los procedimientos de revisión por parte del cliente, la validación de datos, procedimientos de corrección y procesos de conciliación de datos y sistema fuente (ver figura 4.17).

Se debe trabajar muy de cerca con el cliente para crear el plan y las especificaciones de conversión dejando a la experiencia del encargado del proceso, la mejor forma de hacerlo. El plan es revisado y comentado con el cliente para llegar a un punto de acuerdo; de esta manera el cliente sabe a lo que se enfrentará en el futuro próximo y qué debe esperar de los resultados.

Para crear especificaciones de conversión efectivas, se deben de conocer, tanto como sea posible, los datos de origen, incluyendo sus estructuras y el significado de cada uno de sus campos. También se debe de documentar como los datos de la fuente irán alimentando las bases destino. No solo los clientes deben validar el significado de los datos de origen, también deben de estar al tanto de cómo éstos serán relacionados en el warehouse.

El primer lugar para la conciliación de datos es el extraer la información a unos esquemas intermedios que contengan de preferencia datos numéricos como cantidades o valores y ejecutar rutinas que den como resultado la suma o cualquier otra operación existente en el sistema anterior; es lógico pensar que como resultado de la prueba esperemos que los dos valores sean iguales.

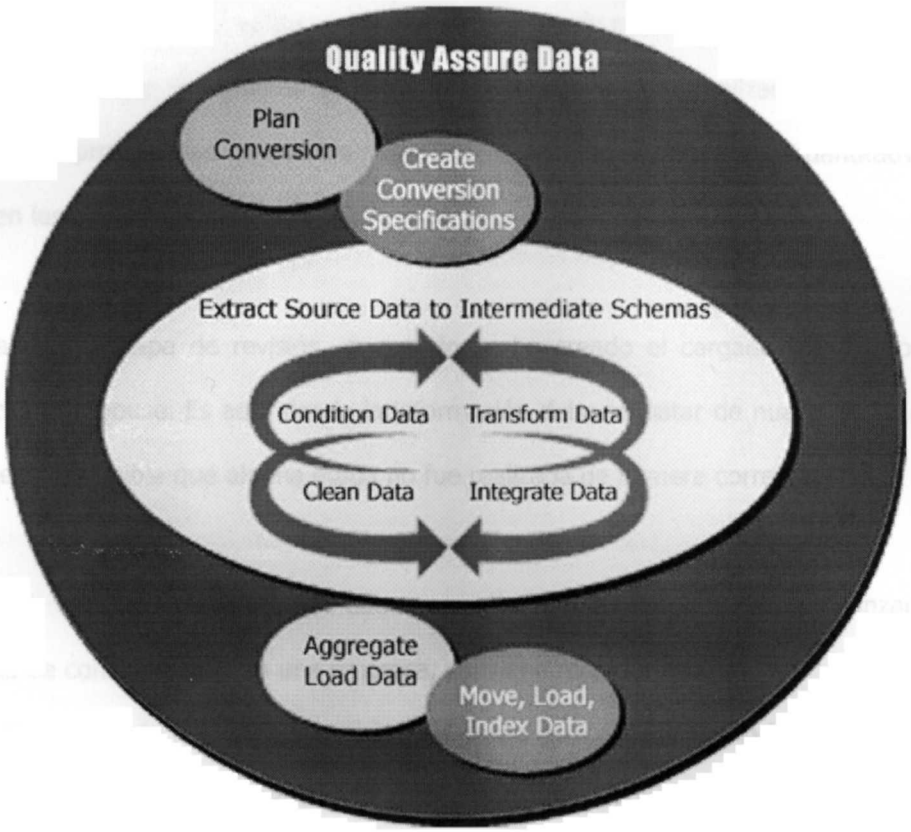


Figura 4.17.El proceso de conversión de datos. © 1997 Miller Freeman, Inc.

Esto se debe dar como regla durante todo el proceso de conversión y entre más pronto comience, será mejor para el proceso en general.

Durante el acondicionamiento, filtrado, transformación e integración de los datos se deben hacer pruebas aleatorias con el cliente para revisar la validez de la información. Cuando se descubren datos erróneos se deben reportar las fallas a los equipos de trabajo para que entre ellos definan en donde se encuentra el error (en los datos fuente o en la conversión), y corregirlo de manera inmediata. Si la corrección se hace en las rutinas de conversión, las correcciones pertinentes se deben hacer también en los metadatos de conversión.

La siguiente fase en donde se hace una conciliación es al momento de pasar los datos del origen al nivel básico de almacenamiento del warehouse; de nuevo es recomendable ejecutar el

mismo procedimiento que se aplico en el paso anterior hasta que ambas partes queden de acuerdo. Se debe recordar que el siguiente paso en la conversión es la actualización definitiva y que ahí existe una alta probabilidad de que los datos ya no concuerden de manera cuantitativa, por ello el hincapié en las dos pruebas anteriores.

La última etapa de revisión es cuando se ha creado el cargador de hechos o máquina analítica del warehouse. Es aquí donde la información debe empatar de nuevo con el origen; de lo contrario es presumible que alguna etapa no fue realizada de manera correcta.

Toda esta tecnología es sin duda muy útil. Sin embargo, antes de lanzarnos a poner "tecnología de conocimiento" en una empresa, conviene recordar que:

- **La tecnología no es mas que un pequeño porcentaje de todo lo que tenemos que hacer para lograr realmente una organización aprendiente**. Se va a requerir un compromiso de la empresa hacia la administración de su capital intelectual. Una administración que involucre identificar el capital, poder localizarlo, estrategias para hacerlo crecer, estrategias para almacenarlo, planes de medida y recompensas para que se reuse ese conocimiento almacenado lo más posible; y más importante que todo eso es el crear un ambiente organizacional que favorezca ese aprendizaje. Además de la memoria organizacional, se requiere un ambiente que promueva el aprendizaje individual, que haga que ese aprendizaje individual se convierta en aprendizaje organizacional a partir de compartir todo ese aprendizaje, todo ello apoyado por el ambiente y auxiliado con la memoria organizacional apoyada por tecnología informática.
- **La memoria organizacional y su tecnología no es más que una parte de lo que involucra la Administración del Conocimiento en una empresa**. En sí, la tecnología se utiliza principalmente para los procesos de clasificar, almacenar y reutilizar conocimiento, únicamente. Recordemos que hay otros 18 procesos del conocimiento. La misión de la Administración de Conocimiento debe ser trabajar en la mejora continua de los

17 procesos. Esto involucra procedimientos administrativos, metodología y tecnología. El campo es muy nuevo y falta mucha investigación por hacer: por ejemplo, en la literatura ni siquiera se habla de procesos del conocimiento. Además, hay en estos momentos confusión en el ambiente entre lo que significa "Administrar el conocimiento" y "lograr una organización aprendiente". En el modelo del Dr. Macquardt, se distingue claramente que la administración del conocimiento es solamente una parte de la OA.

- Por último, **nunca debemos olvidar que la tecnología en sí no es ninguna solución a nada**. Considera este comentario del consultor Ricardo Díaz de Cemex: "Las intranets son una nueva forma de 'poner a disposición la Información', pero como podemos observar en Internet, esta nueva forma no le da valor por si sola. Son las personas detrás de las herramientas las que organizan, filtran, interpretan y le dan valor convirtiéndola en información útil al negocio". Lo mismo podríamos decir del conocimiento, y de cualquier tecnología que se utilice para almacenarlo.

8. Potencial para el aprendizaje más rápido

Un aspecto decisivo del rápido aprendizaje es buscar el contenido y los métodos "apropiados". Los empleados se deben hacer las siguientes preguntas: ¿Nos estamos enfocando en lo esencial qué debemos aprender?, ¿Vamos aprendiendo en la forma apropiada?.

Hay cuatro tipos de organizaciones que tienen el mayor potencial para convertirse en organizaciones de más rápido aprendizaje:

1. **Las de terreno virgen**: aquellas que empiezan de la nada, sin ninguna cultura organizacional que las agobie.
2. **Las industrias de ritmo rápido**: aquellas en las que el aprendizaje rápido es decisivo para su supervivencia (por ejemplo, las compañías de hardware/software para computadoras).

3. **Las que son líderes en su ramo**: aquellas que se enorgullecen de su reputación de estar a la vanguardia.
4. **Las que van en decadencia**: aquellas que han sufrido una pérdida traumática de su ventaja competitiva y que ha tenido que luchar para encontrar nuevas formas de hacer negocios.

En la actualidad, la mayoría de las instituciones a atravesado por alguna suerte de examen de conciencia acerca de la forma en la que realizan negocios. Y casi todas han decidido que ya no deben operar como antes lo hacían. La seriedad con que aborden este problema determinará su potencial como organizaciones de rápido aprendizaje.

Cuando les preguntan por qué sus empresas deben aprender, los gerentes responden que la cuestión es muy sencilla: aprender o morir y narran ejemplos de haber perdido negocios debido a que un competidor los superó en el aprendizaje.

CAPITULO V

DESARROLLO DEL EXPERIMENTO

El objetivo de esta tesis es demostrar que el uso de bases de conocimiento ayuda a los departamentos de desarrollo y soporte de sistemas, a disminuir el tiempo de respuesta a los usuarios, cuando se presentan fallas tanto en el software como en el hardware. En el lapso que duró la recolección de la muestra, se trabajó con dos variables: **tiempo de respuesta y tiempo entre fallas**. Estos dos indicadores son fundamentales para medir el desempeño de las áreas de informática e inclusive, han llegado a ser considerados por algunas empresas como indicadores de calidad y hasta core value[1].

Este tipo de trabajo no debería de sorprender a una empresa que aplica técnicas de calidad en el software o inclusive, técnicas de auditoría para la recuperación de errores; sin embargo, lamentablemente sólo una de las empresas de estudio lleva a cabo un proceso a medias de la recuperación de errores. Para la generalidad, la auditoría y la calidad del software son elementos que están más allá de sus necesidades como departamento e inclusive como empresa; debido a que están más preocupadas por certificarse en sus áreas de producción y no en las de apoyo. Incluso, al momento de analizar la documentación de los sistemas (cuando existía) se observó que en el aspecto de recuperación de errores, los sistemas no contaban con rutinas que pudiesen ayudar a una pronta detección, recuperación y administración de los fallos, es decir, en muchas ocasiones, al ocurrir un problema, el usuario debía de esperar hasta que la persona indicada acudiera a su terminal, examinara el mensaje de error que le había enviado el sistema, anotarlo, aplicar su procedimiento para recuperar el error y por último dirigirse al código fuente, buscar la falla y corregirla. Este además de ser un procedimiento que consume mucho tiempo, tanto para el

---

[1] Core value. Valor agregado que las empresas dan a los clientes a través de sus productos y/o servicios.

usuario como para el programador, no cumple con los procedimientos básicos de auditoría ni de desarrollo de software (bajo el enfoque de calidad), en el que se deben desarrollar rutinas que indiquen claramente el lugar en dónde se generó el error, qué variables estaban en memoria, características del equipo en dónde se trabajaba, error generado, fecha y hora. Este es uno de los problemas más comunes que tienen las empresas que se dedican a la elaboración de software o que desarrollan su propio software y como se mencionó en el capítulo 3, es donde van a parar más del 50% de los recursos empleados para un software nuevo.

Para la toma de datos, el primer paso a seguir fue el recolectar la información necesaria para mostrar que los tiempos de respuesta de las áreas de soporte/desarrollo disminuyen después de detectada y reportada una falla en el software; sí es que se documentan correctamente todas las fases del desarrollo y mantenimiento de software (investigación, corrección, rectificación y puesta en marcha de la nueva versión). En el segundo paso, se utilizó el tiempo entre fallas (del mismo tipo) para verificar que las ocurrencias entre éstas son más espaciadas o nulas.

Cabe hacer mención que se realizó una preferencia por aquellos sistemas o módulos que fueron atendidos (en sus diferentes etapas de desarrollo y mantenimiento) por una sola persona y con ello se da la posibilidad de reducir el sesgo que se produciría en la muestra por el problema de los diferentes niveles de habilidades que pudiesen existir en un grupo de programadores, aunque también resulta importante mostrar el efecto de los grupos de trabajo entendidos como un solo ente. En este capítulo se explicará la obtención de datos, los métodos empleados para el análisis de la información y algunas interpretaciones.

1. Base de conocimientos

Este fue un punto en el que las cinco empresas concordaron en llevar (o ya tenían) una forma más o menos parecida en registrar su información en carpetas (empresa dos) o en medios

electrónicos (empresas restantes). Toda la documentación contenida en las carpetas y en los medios electrónicos maneja una estructura de árbol para facilitar su manejo, incluyendo referencias cruzadas a otros temas relacionados con el problema original.

Aunque el concepto de administración de la información es el mismo para cualquier medio de almacenamiento masivo, llámese papel, cinta o disco, no es así su manejo, mantenimiento y el concepto que se debe de tener para operarlo; no es lo mismo hacer un manual de errores que tenga un índice y una referencia cruzada, que el manejar un software como con el que contaban las empresas cuatro y cinco; éste era un software de fabricación propia fundamentado de bases de datos que tenían como llave un conjunto de palabras clave que describían su contenido. Una herramienta un poco más sofisticada fue el empleado en las empresas uno y tres, que a través de un navegador de internet les ayuda a administrar un "foro" de discusión y en él manejar toda la información que se ha ido almacenando con el tiempo y que además, les brinda las facilidades de las ligas a otros tópicos relacionados, incluso fuera de su dominio (o lugar físico). El software empleado en las empresas fue el hypernews[2], a continuación se presenta una breve descripción hecha por el autor de ésta herramienta:

"HyperNews is a cross between the hypermedia of the WWW and Usenet News. Readers can browse through the messages written by other people and reply to those messages. A forum (also called a base article) holds a tree of these messages, displayed as an indented outline that shows how the messages are related (i.e. all replies to a message are listed under it and indented). Users can become members of HyperNews or subscribe to a forum in order to get email whenever a message is posted, so they don't have to check if anything new has been added. A recipient can then send a reply email back to HyperNews, rather than finding a browser to write a reply, and HyperNews then places the message in the appropriate forum."[3]

---

[2] Se puede encontrar más información sobre el software en: http://www.hypernews.org/, 10.11.1998.
[3] LaLiberte, Daniel About Hypernews http://www.hypernews.org/HyperNews/get/hypernews/about.html, 10.11.1998.

Como se podrá entender, este foro electrónico les ayudo a organizar sus documentos de tal manera que, una vez "iniciado" el foro, todos los que estén suscritos a él pueden actualizar la información que se encuentra almacenada (modificarla o agregar nuevo conocimiento); en cambio, los que no están habilitados para participar en él, pueden acceder al sistema con el único privilegio de leer los documentos. Es de suponer que la administración de la información es labor de algunos cuantos y que la gran mayoría de los empleados solo participa aportando conocimiento o en su defecto, recibiéndolo; esto depende de las políticas establecidas en la empresa sobre la administración del conocimiento y de los accesos a la misma.

2. Recolección y análisis de la muestra

Toluca es una región cuya economía se basa fuertemente en la industria manufacturera y de alimentos; las áreas de servicios no han sido tan desarrolladas debido a que la Ciudad de México se encuentra a menos de 60 Km y ahí se encuentra uno de los principales proveedores de este tipo de bienes por lo que no fue posible realizar un análisis directo con una compañía de consultoría en informática; en cambio las empresas que decidieron participar contaban con un departamento que tenía al menos las funciones de mantenimiento de los sistemas (una de las empresas contaba con una sola persona para tal labor).

Las bases de datos de análisis, en primer lugar, fueron facilitadas por los departamentos de sistemas de cuatro empresas situadas en el corredor industrial Toluca-Lerma y Toluca-Atlacomulco. Cuatro de ellas pertenecientes al sector manufacturero:

1. **AUMA Lerma, S.A. de C.V.** Dedicada a la fabricación de piezas de aluminio para el área automotriz. Empresa uno.
2. **Manufacturas Textiles Ideal, S.A. de C.V.** Empresa dedicada a la fabricación de cintas para la industria textil y cinturones de seguridad. Empresa dos.

3. **Grupo IUSA.** De este grupo se trabajo con el área de conductores eléctricos. Empresa cuatro.
4. **Pfizer, S.A. de C.V.** Empresa farmacéutica, que entre algunos de sus últimos logros esta la fabricación de la pastilla "viagra". Empresa cinco.

Y en segundo lugar por **el departamento de sistemas del Tecnológico de Monterrey Campus Toluca** (empresa tres). Inicialmente se tenían contempladas más empresas para la recolección de información pero, lamentablemente, la colaboración de los empleados encargados de capturar la información en las hojas de registro de eventos de las propias áreas fue muy poca y hasta nula en algunos casos (como mencioné anteriormente, el proyecto pareció importante para muchas empresas, solo que estos indicadores dañaban la imagen de los empleados y por ello no cooperaron como se debía).

De las empresas que aceptaron participar en la investigación se seleccionaron aquellos proyectos que contaban con la suficiente información de tiempo y tipo de fallas, la empresa uno participó con muestras de los sistemas de personal (nómina) y cuentas por pagar; la empresa dos, aporto datos de un sistema de implosión de materiales; la tercera empresa aporta muestras del área de contabilidad, la empresa cuatro también se relacionó con el área de materiales y la empresa cinco aporto información de un sistema de ventas.

Durante la recolección de datos se presentó el problema de las diferentes plataformas con las que se opera, además de los diversos tipos de aplicaciones y lenguajes de programación. Para facilitar la clasificación de la información y su manejo, se decidió estandarizar los números de error y sus descripciones a los que se manejan en el FoxPro de Microsoft. En la tabla número 1 se hace un comparativo de las plataformas de trabajo de las empresas.

	Sistema operativo	Manejador de bases de datos	Aplicación
Auma	SCO Unix	Informix	Nómina, CXP
MATISA	Novell	Clipper	Materiales
Tec. Toluca	Windows NT	FoxPro	Contabilidad
IUSA	HP-UX	Progress	Materiales
Pfizer	AS400	DB400	Ventas

Tabla 5.1. Muestra de las diferentes plataformas de trabajo.

Como se observa en la tabla anterior, hubiese resultado incómodo trabajar con cinco diferentes tipos de administradores de errores; si bien es cierto que todos manejan el concepto de trabajo en red, archivo, variable y distribución de recursos, sus clasificaciones y nomenclaturas de los errores varían de manera considerable. Sólo tomemos en cuenta que tanto Clipper como FoxPro comienzan su nomenclatura de los errores desde el número uno y crece hasta cuatro dígitos y no tienen un agrupamiento especial por familia de errores (errores de red, variable, memoria, etc.); en cambio, la nomenclatura utilizada en Informix es de cuatro caracteres, además de estar agrupados por tipo de error (de archivo, de aplicación, etc.) y construir algunos basándose en el sistema operativo sobre el que está instalado (cabe mencionar que los mensajes de error de un SCO Unix son diferentes de un HP-UX o AIX, dependiendo también de la versión que se opere).

Enseguida se muestran los tipos de errores y sus incidencias por cada uno de los sistemas que se probaron. En el tiempo que duró la recolección de datos se distinguió una fecha en especial: 1º de julio, ya que a partir de ahí, se considera que todos los sistemas a tratar ya contaban con un sistema de administración del conocimiento y se aplicaba ésta técnica para la solución de errores.

El tamaño de las muestras varía, primero, por los diferentes niveles de habilidad de los programadores y gentes de soporte de las empresas y en segundo, por el tiempo que se tardó el

departamento en implantar los mecanismos tanto para la recolección de información (es decir, una herramienta en donde se llevan los tipos de fallas, quien lo atendió y cuanto tiempo tardo) como para la administración del conocimiento y su aplicación. Lo anterior fue un trabajo algo desigual, esto debido a que en unas empresas ya se tenía la concepción de lo que se necesitaba y, de hecho, ya trabajaba con ello; y en otras todavía no se tenía pensado llevar este tipo de "bitácoras" y por supuesto que en éstas se tuvo que platicar con el personal encargado cuál era el origen y utilidad de las mismas. Como se indica en la siguiente tabla, las muestras fueron:

Sistema	Muestra total
Nómina	413
Cuentas por Pagar	157
Materiales (MATISA)	196
Contabilidad	107
Materiales (IUSA)	39
Ventas	88

Tabla 5.2. Tamaño de las muestras.

A continuación se muestran las gráficas de la distribución por error de los diferentes sistemas.

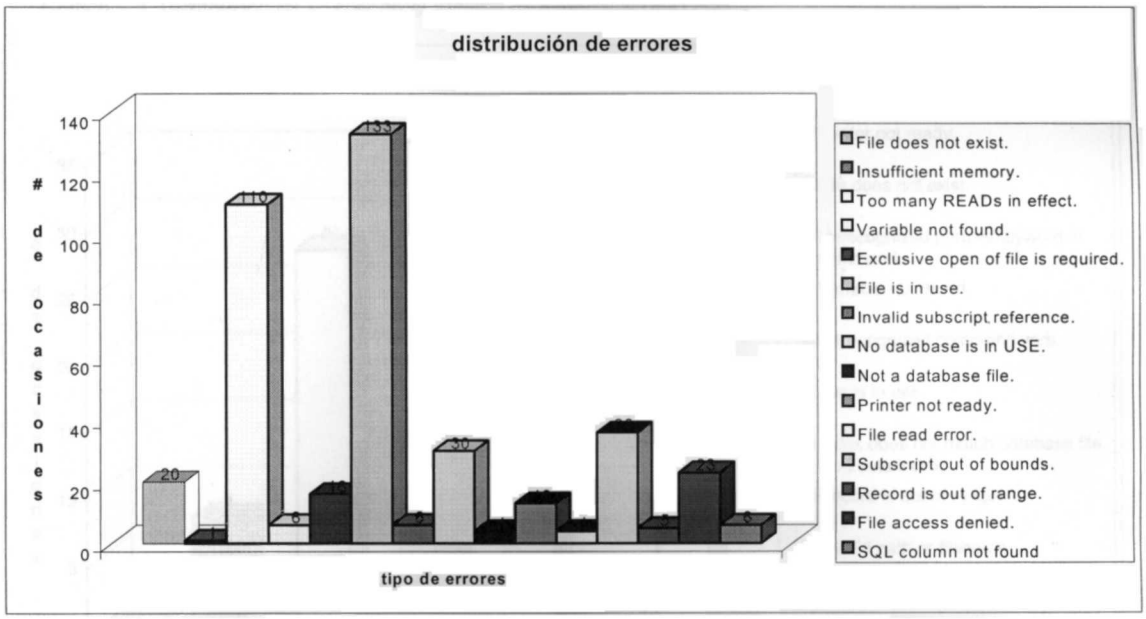


Gráfico 5.1. Distribución de errores en el sistema de nómina.

Gráfico 5.2. Distribución de errores del sistema de cuentas por pagar.

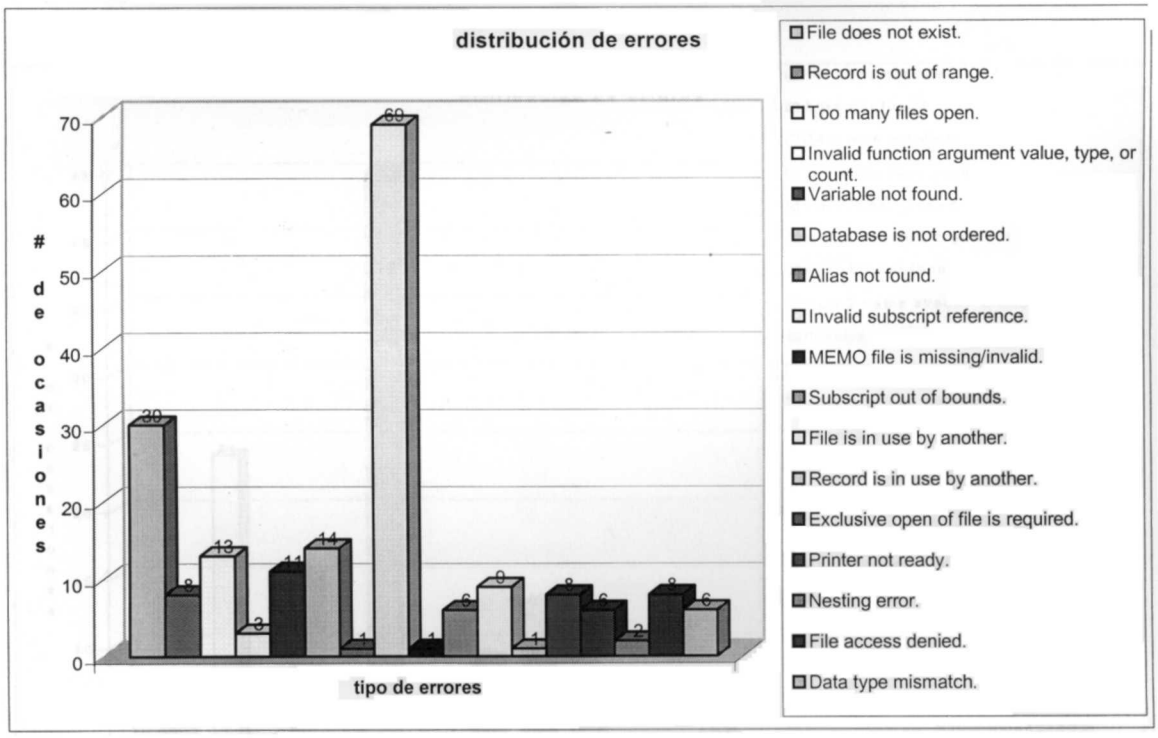


Gráfico 5.3. Distribución de errores en el sistema de materiales (MATISA).

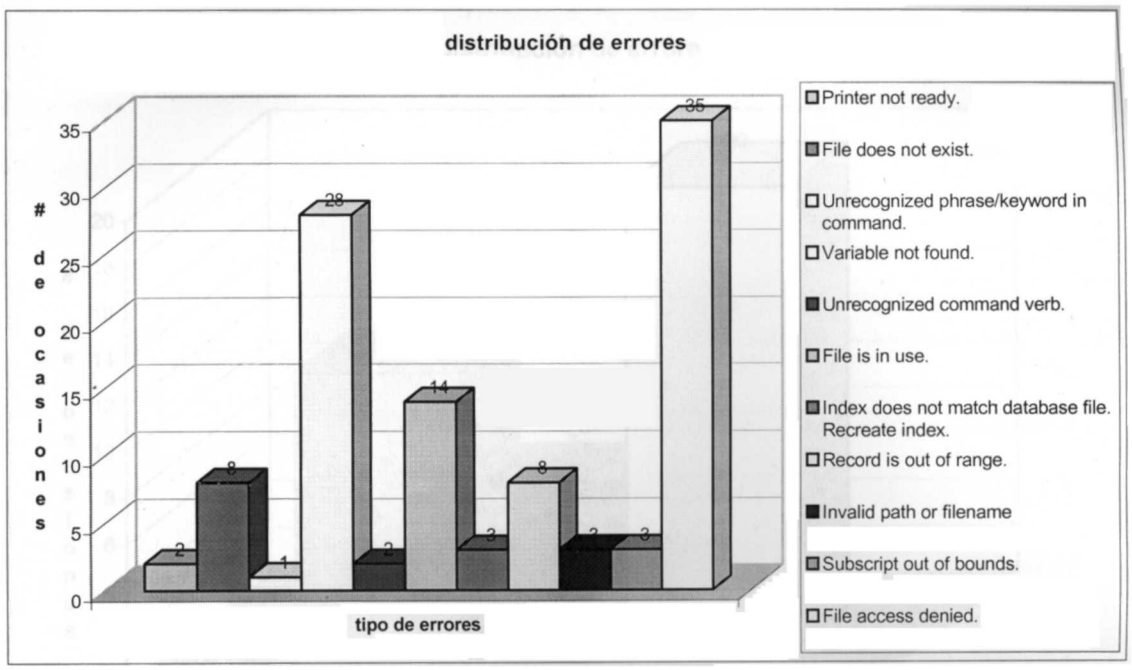


Gráfico 5.4. Distribución de errores del sistema de contabilidad.

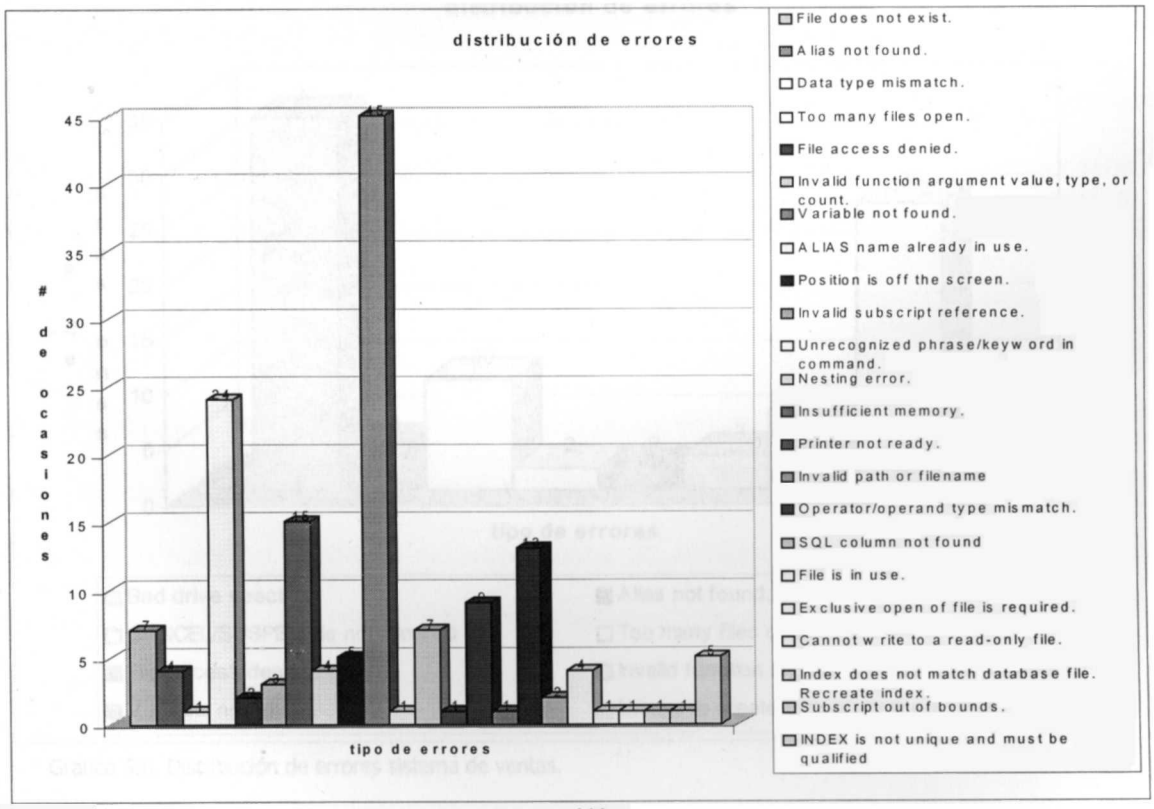
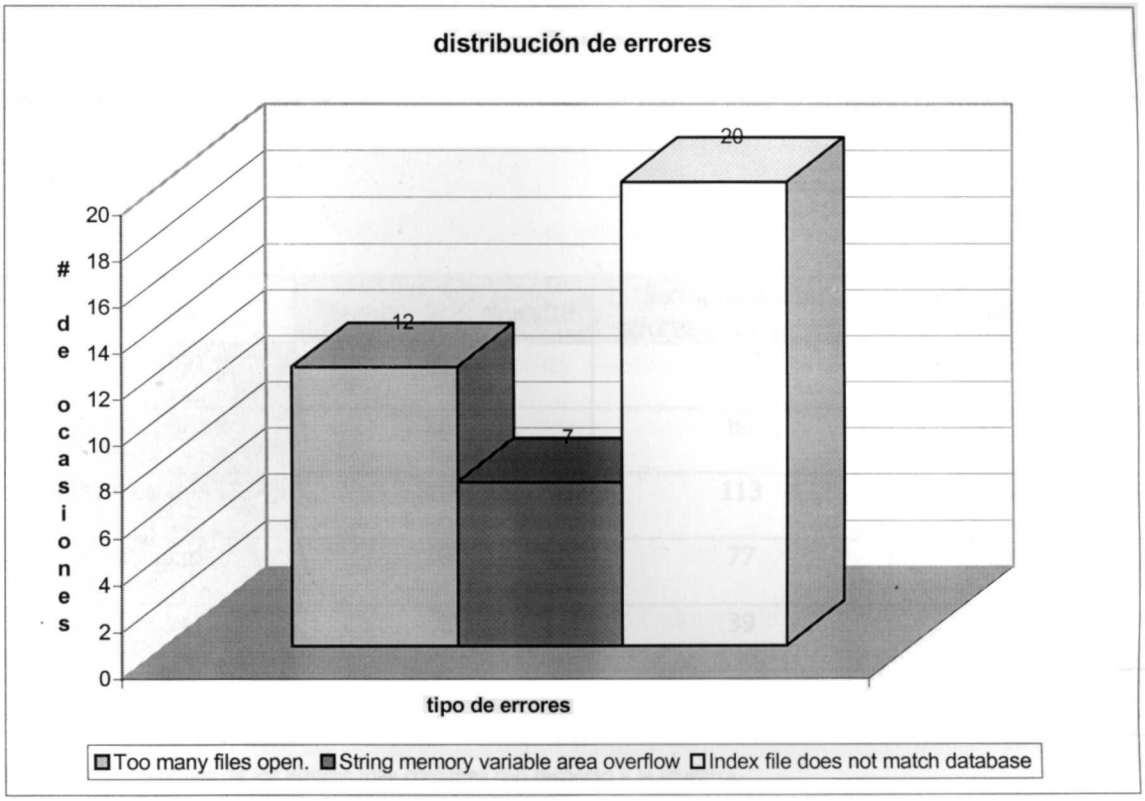


Gráfico 5.5. Distribución de errores del sistema de materiales (IUSA)



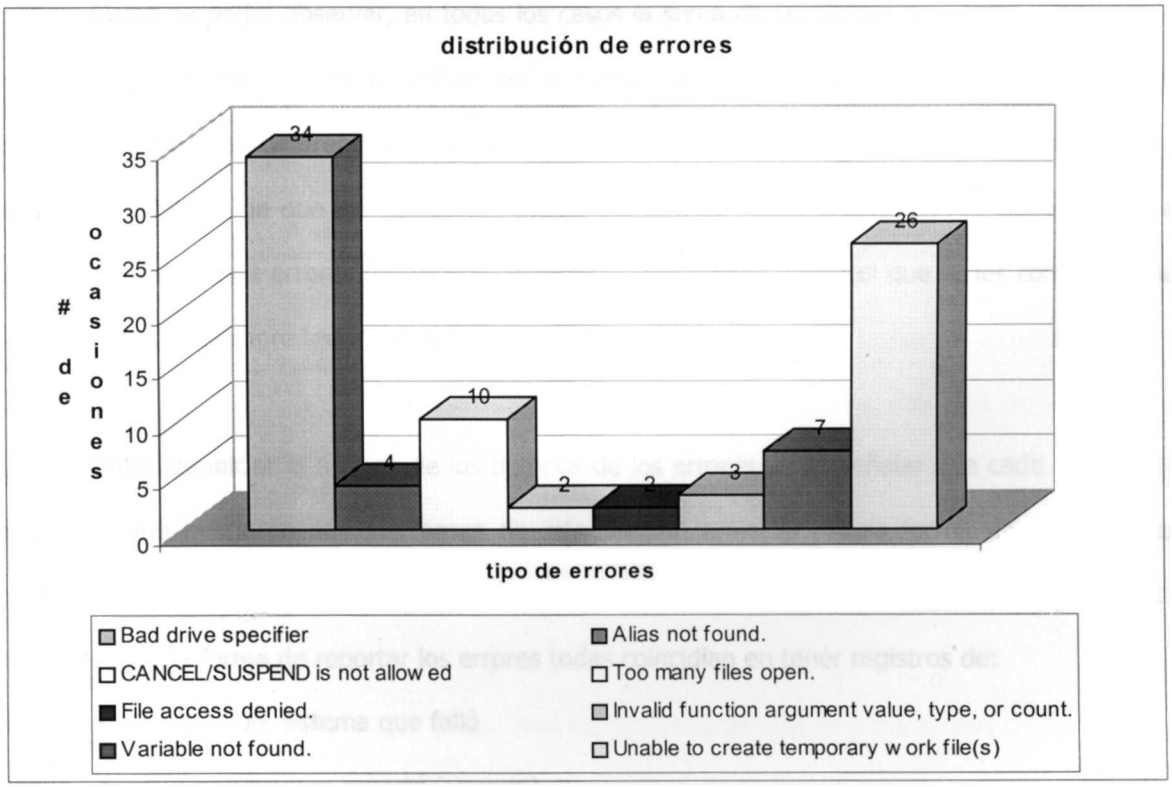


Gráfico 5.6. Distribución de errores sistema de ventas.

Para acotar el estudio y la cantidad de errores, se tomaron de cada sistema los tres errores con mayor número de incidencias; al trabajar con ellos, podremos lograr un análisis más puntual de los diferentes tiempos de respuesta de cada uno. Se llego a esta conclusión después de analizar la siguiente tabla

Sistema	Tamaño de la muestra	Suma de los tres errores más comunes	Porcentaje
Nómina	413	279	**67.55%**
Cuentas por pagar	157	84	**53.50%**
Materiales (MATISA)	196	113	**57.65%**
Contabilidad	107	77	**71.96%**
Materiales (IUSA)	39	39	**100.00%**
Pfizer	88	70	**87.50%**

Tabla 5.3. Porcentajes de los errores más comunes con respecto a la muestra.

Como se podrá observar, en todos los casos la suma de las incidencias de los errores más comunes representa más de la mitad del universo; asimismo éstos cuentan con una mayor probabilidad de tener ocurrencias en el tiempo que duró la recolección de información, en cambio, existe la posibilidad de que los que tienen menor número de sucesos sesguen el estudio debido a que puede tratarse de errores únicos y no existiría un parámetro contra el que se les compare y su serie de tiempos siempre tendería a cero.

Antes de iniciar el análisis de los tiempos de los errores, debo señalar que cada una de las empresas que ayudaron con sus bases de información, tenía su propia forma de registrar los eventos que sucedían y además su propia forma de administrar su conocimiento. Aunque por ejemplo, para la forma de reportar los errores todas coincidían en tener registros de:

- Módulo del sistema que falló
- Nombre del que reportó (usuario)

- Hora y fecha del reporte
- Tipo de error detectado
- Hora y fecha en que terminó la reparación

Un punto de vital importancia para el desarrollo del tema es el referido al tipo de error detectado. ¿Por qué? Porque basándose en él, se va construyendo el registro de conocimientos que ayudará al departamento. Y esto **si fue algo común en todas las empresas**: sin excepción alguna, utilizaron como llave el tipo/número de error para crear sus bases de conocimientos.

A continuación se muestran las gráficas de los tres errores más comunes por sistema, el tiempo que tardaron en ser atendidos y su incidencia (número de ocasiones en que se presentaron) durante el tiempo de recolección de información; que en el caso más antiguo fue del mes de junio de 1997 y hasta el mes de octubre de 1998. La manera de presentar la información corresponde, primero al tiempo en segundos que tardo el encargado de los sistemas en atender a los usuarios y corregir la falla; estas gráficas tienen el formato de líneas y sirven, en este caso, para mostrarnos la tendencia en el tópico de tiempo de respuesta y como éste fue evolucionando para sus diferentes ocurrencias. Debemos aclarar que para éste análisis no nos importa cuándo sucedieron las fallas (para los fines de esta investigación, se esperaría que la tendencia de todas las líneas sea de mayor a menor en el eje del tiempo de respuesta) y la segunda gráfica, mostrada en puntos, nos indica el número de errores que se reportaron en una fecha en específico y su variación en el tiempo; de nuevo, para los fines que persigue esta inspección se espera que entre más nos acerquemos al presente, el número de incidencias de cada uno de los eventos se reduzca, en otras palabras, se espera que la gráfica se cargue del lado izquierdo; ello nos indicaría que el número de ocurrencias de los errores van desapareciendo según avanzamos al presente.

De acuerdo a lo planteado anteriormente, la unión de ambas gráficas nos debería indicar que según transcurre el tiempo, el número de errores reportados es menor (respuesta lógica si

tomamos en consideración que se están eliminando los errores del sistema) y que el encargado de los sistemas responde con mayor eficiencia a los problemas (disminución en el plazo de respuesta); atacando el problema, ayudado por el almacenamiento del contexto del sistema en operación y la información que se ha recopilado sobre los errores de software y hardware.

Gráfico 5.7. Comportamiento de los tres principales errores del sistema de nómina.

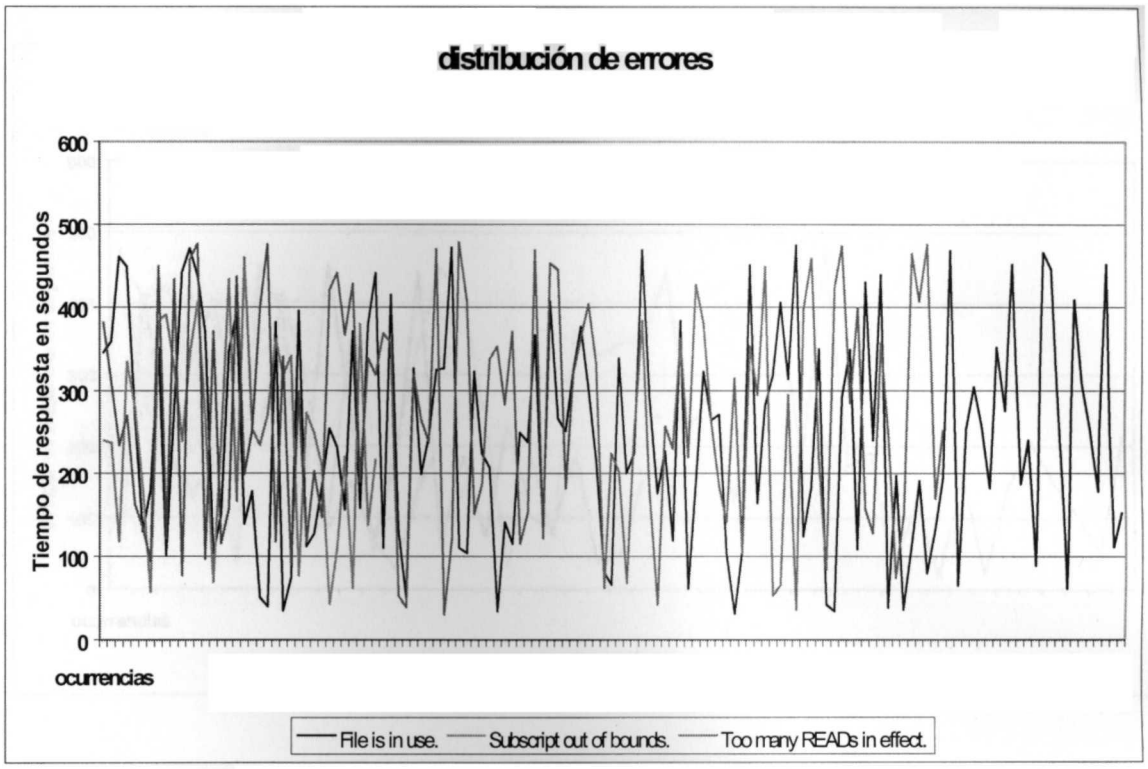


Gráfico 5.8. Fechas en que ocurrieron los tres principales errores del sistema de nómina y sus incidencias.

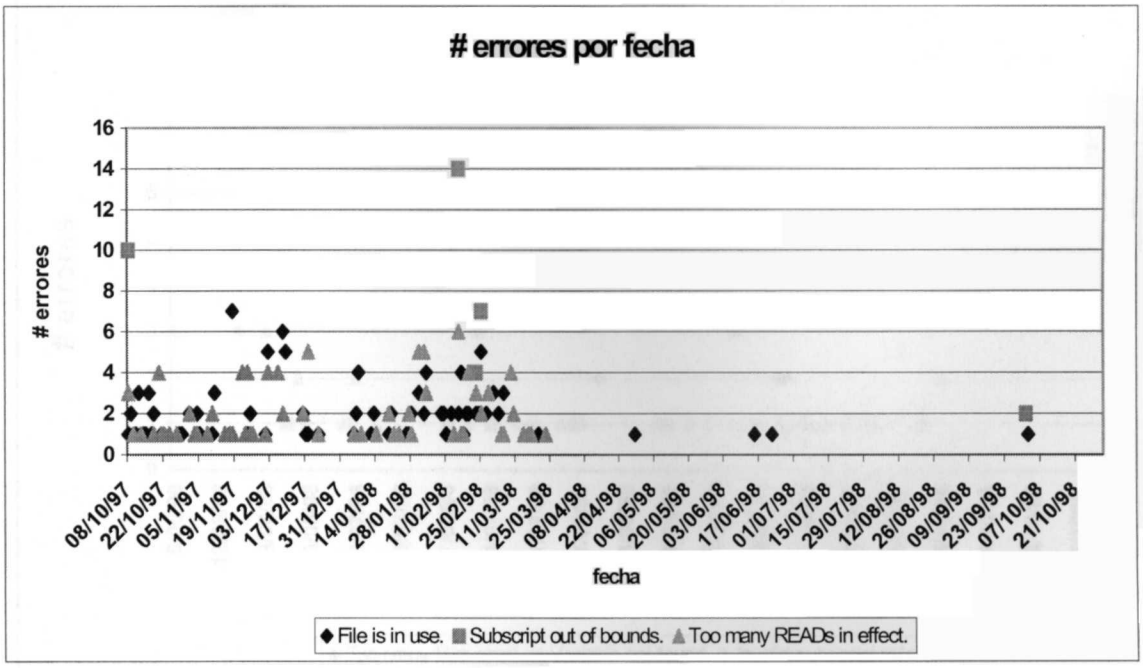
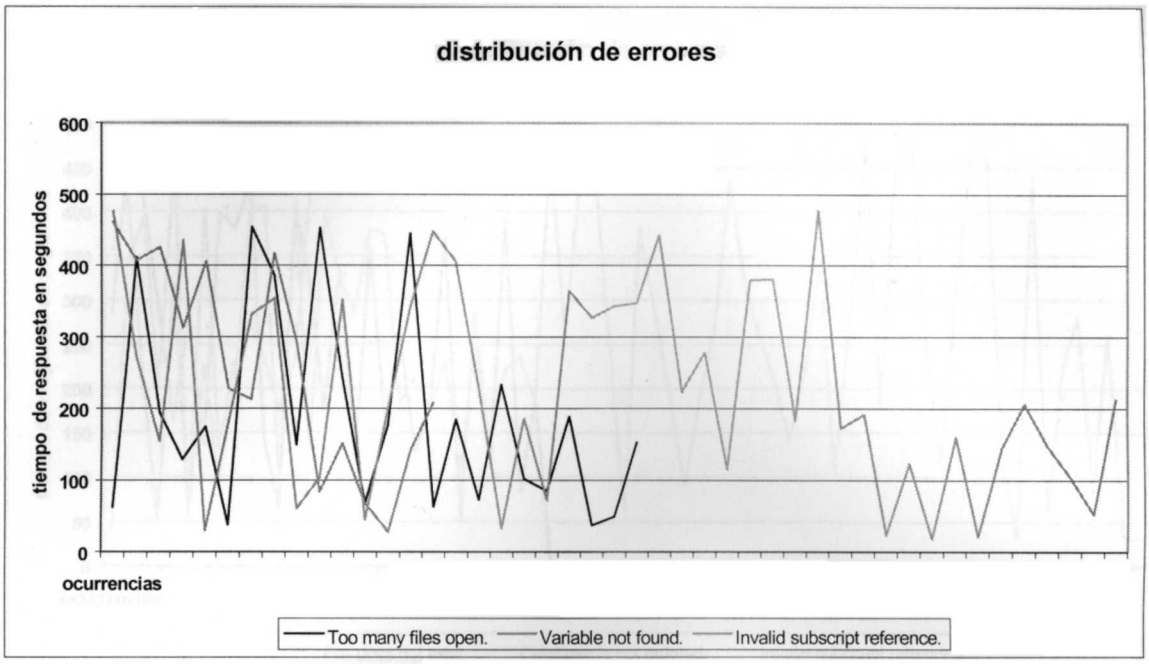


Gráfico 5.9. Comportamiento de los tres principales errores del sistema de cuentas por pagar.



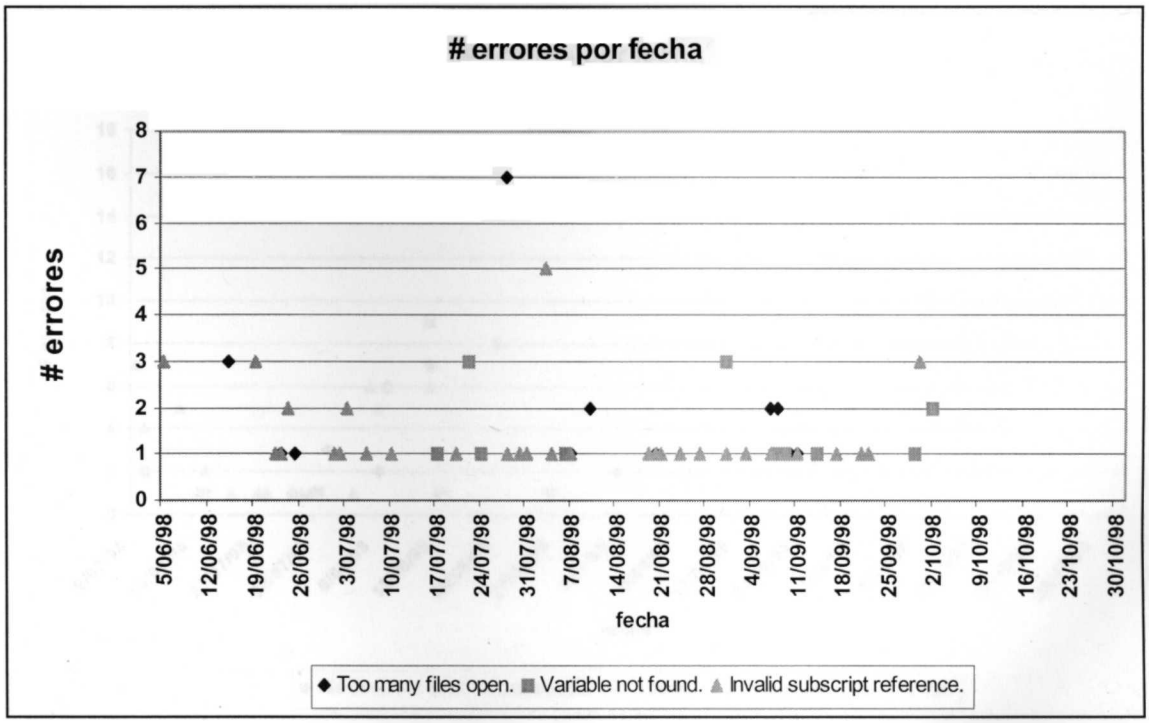
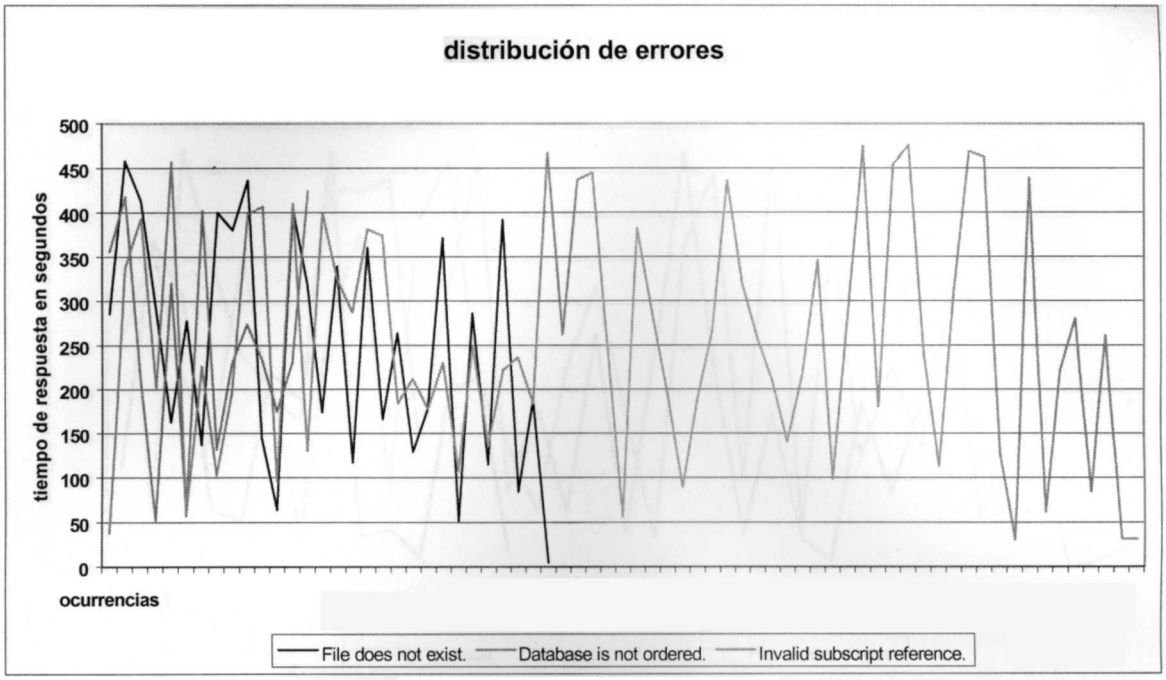


Gráfico 5.10. Fechas en que ocurrieron los tres principales errores del sistema de cuentas por pagar y sus incidencias.

Gráfico 5.11. Comportamiento de los tres principales errores del sistema de materiales (MATISA).



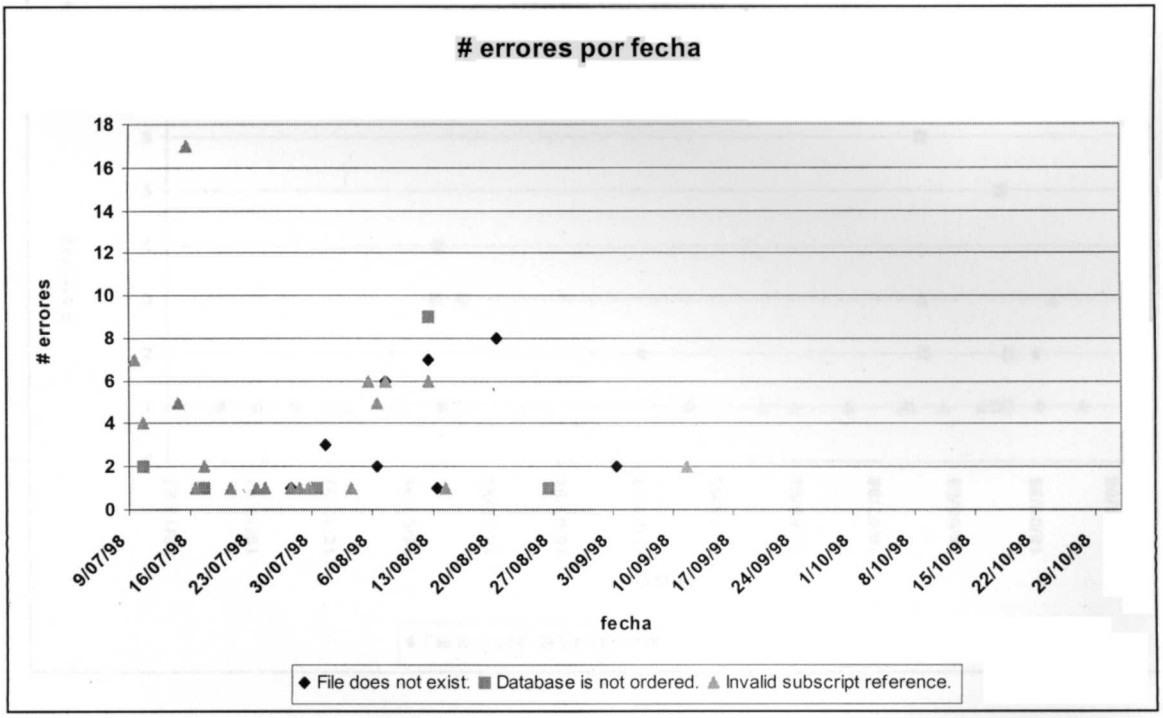
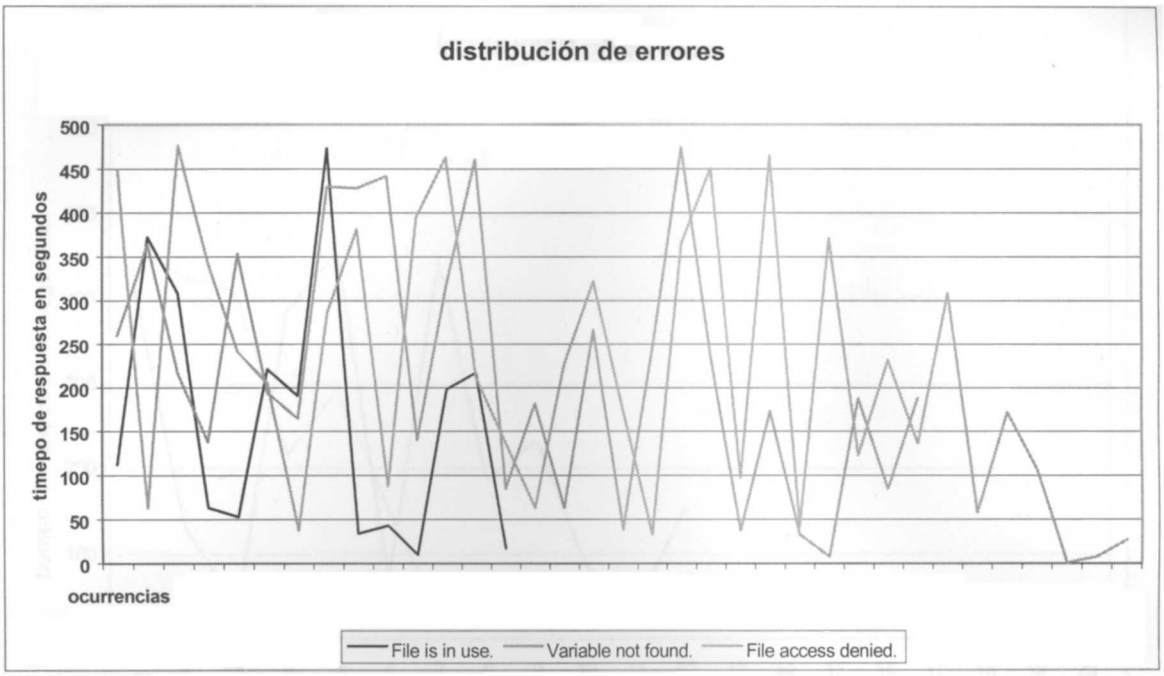


Gráfico 5.12. Fechas en que ocurrieron los tres principales errores del sistema de materiales y sus incidencias (MATISA).

Gráfico 5.13. Comportamiento de los tres principales errores del sistema de contabilidad.



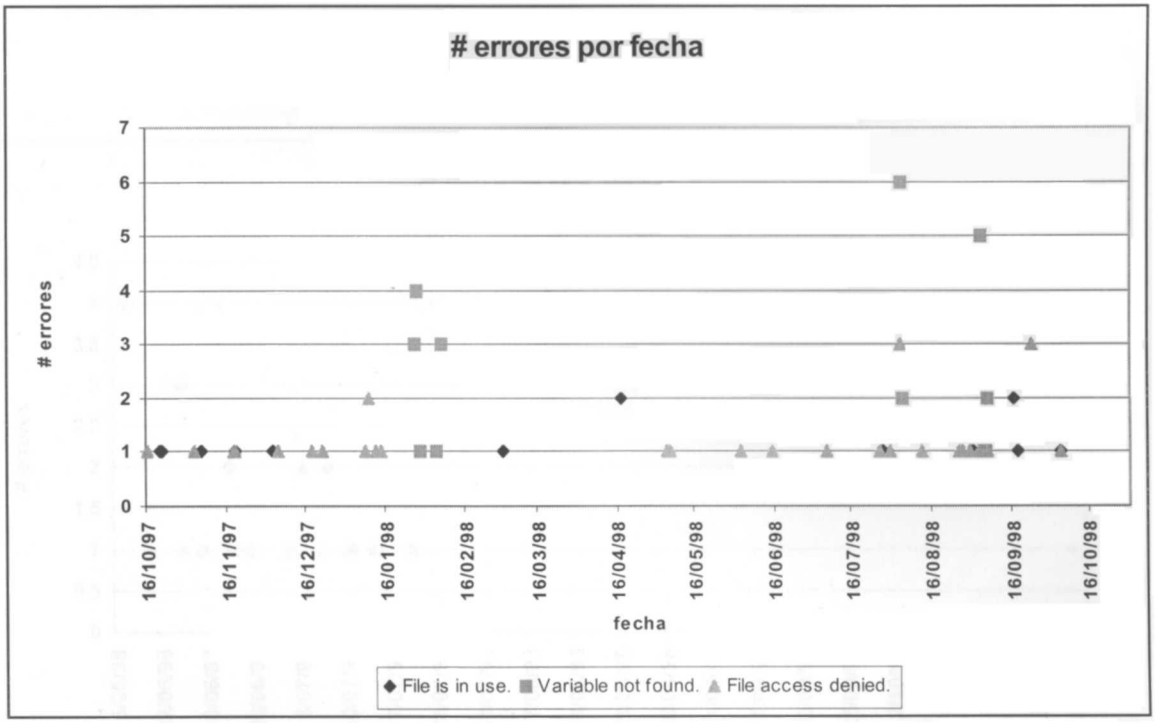


Gráfico 5.14. Fechas en que ocurrieron los tres principales errores del sistema de contabilidad y sus incidencias.

Gráfico 5.15. Comportamiento de los tres principales errores del sistema de materiales (IUSA).

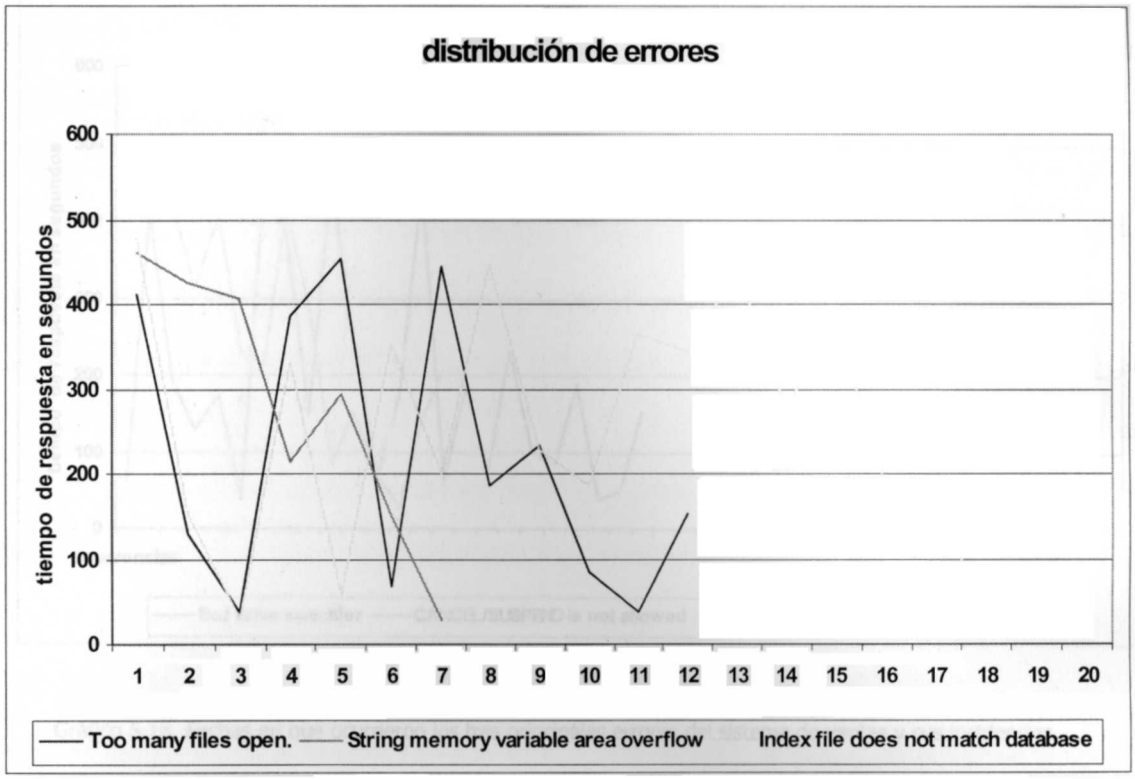


Gráfico 5.16. Fechas en que ocurrieron los tres principales errores del sistema de materiales y sus incidencias (IUSA).

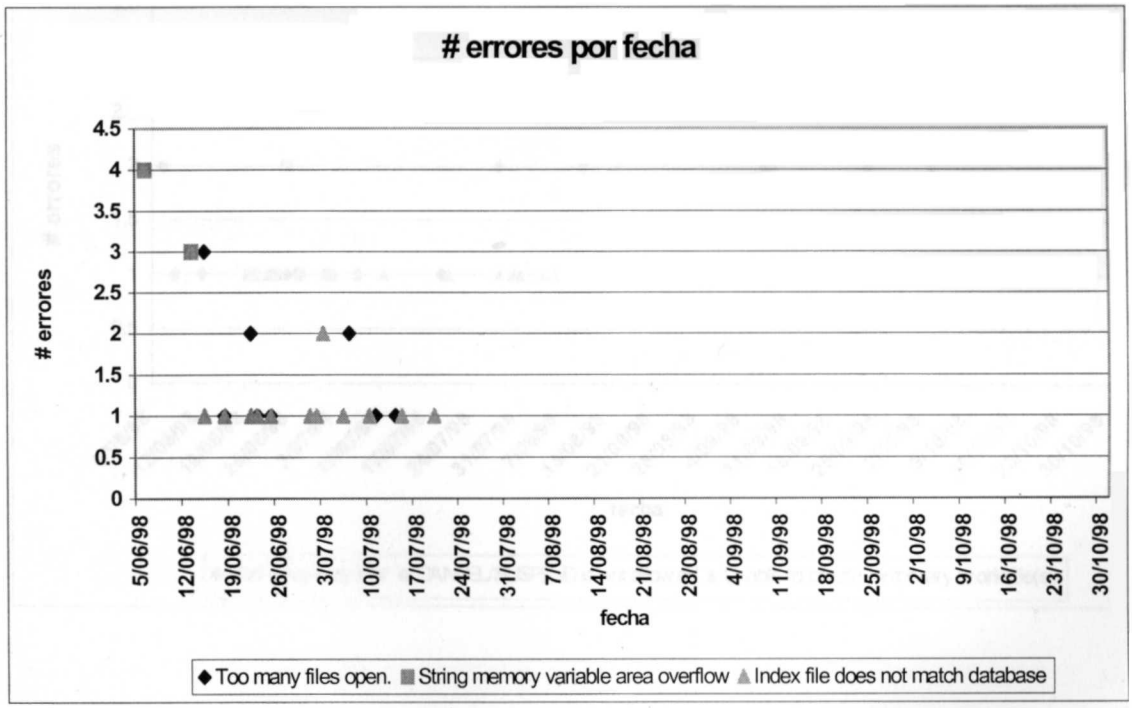


Gráfico 5.17. Comportamiento de los tres principales errores del sistema de ventas.

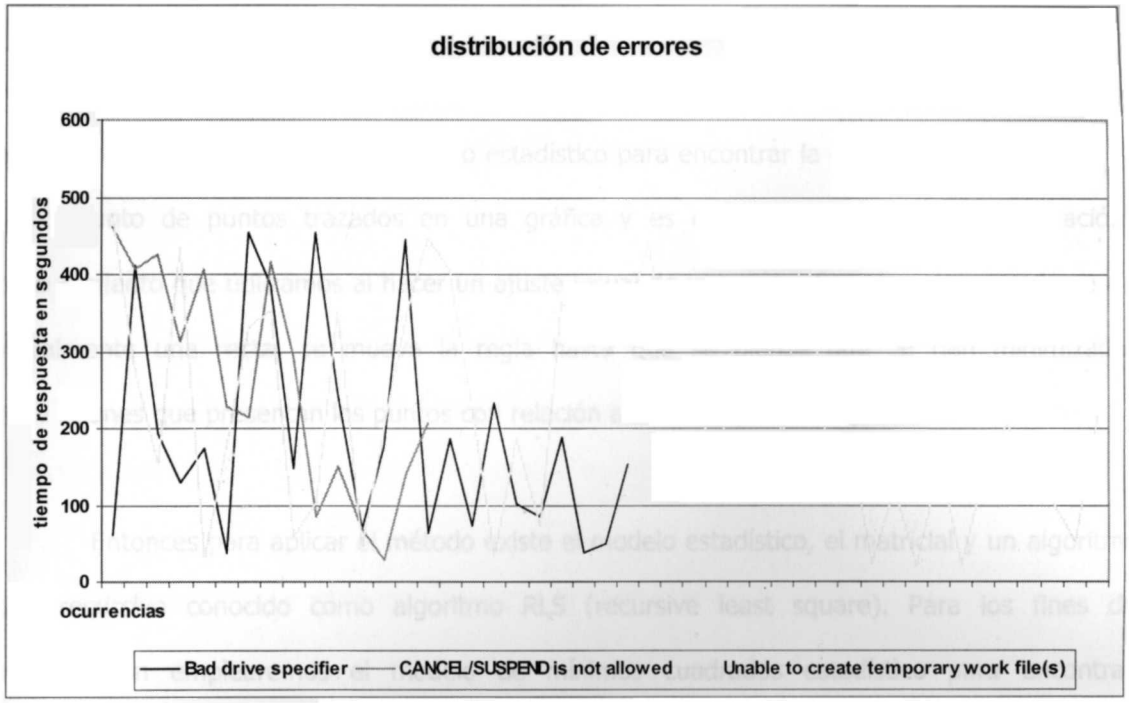
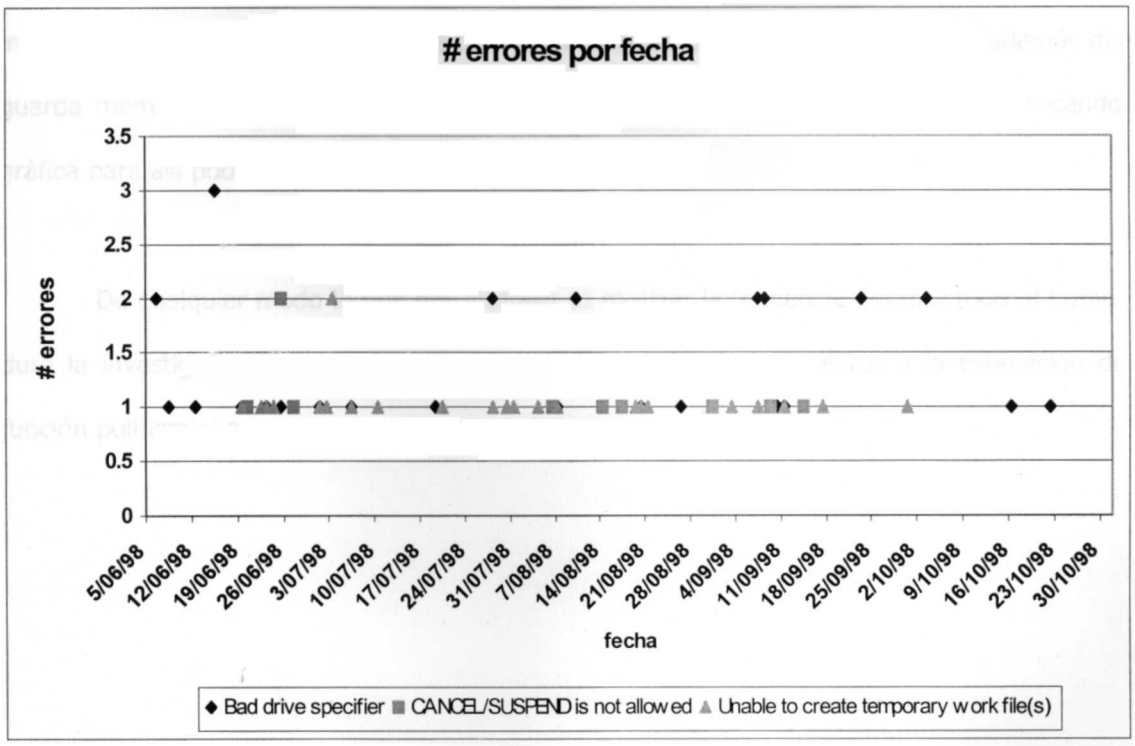


Gráfico 5.18. Fechas en que ocurrieron los tres principales errores del sistema de ventas y sus incidencias.



3. Ajuste de los datos a una recta

Para describir las gráficas con mayor precisión existe un método matemático que ayudará a realizar un mejor análisis de las muestras; éste es el método de mínimos cuadrados. El método de mínimos cuadrados es un procedimiento estadístico para encontrar la línea recta de mejor ajuste a un conjunto de puntos trazados en una gráfica y es en cierto modo, una formalización del procedimiento que utilizamos al hacer un ajuste visual de una recta. Por ejemplo, cuando se ajusta visualmente una recta, se mueve la regla hasta que se piensa que se han minimizado las desviaciones que presentan los puntos con relación a la recta que se propone.

Entonces para aplicar el método existe el modelo estadístico, el matricial y un algoritmo de tipo recursivo conocido como algoritmo RLS (recursive least square). Para los fines de la investigación emplearemos el modelo de mínimos cuadrados estadístico para encontrar la pendiente de las líneas de ajuste (gráficas 5.7, 5.9, 5.11, 5.13, 5.15 y 5.17) de los diferentes errores debido a que el modelo matricial es solo una variación al estadístico y el RLS es un modelo matemático empleado principalmente para filtrar señales electrónicas y de sonido, además que "no guarda memoria" y solo le interesa el ajuste de los nuevos puntos que se van agregando a la gráfica para así poder predecir su comportamiento.

De cualquier modo lo que nos interesa es mostrar la tendencia durante todo el tiempo que duro la investigación y no nos interesa la predicción de nuevos valores o la estimación de una función polinomial que cubra las diferentes observaciones.

El método de mínimos cuadrados asume que: "La pendiente de una recta (B) **se puede** interpretar como una medida del cambio en y por unidad de cambio en x, cuando nos movemos de izquierda a derecha sobre la recta. Si B > 0, el cambio en y es positivo y la línea se desliza hacia arriba a la derecha. Si B < 0, la pendiente es negativa y la recta se desliza hacia abajo a la derecha"[4]

¿Qué podemos interpretar de ello para nuestro experimento? Que una pendiente negativa nos muestra una baja en el tiempo de respuesta según transcurre el tiempo y una positiva es indicativo de que la respuesta tarda más. Esta herramienta nos ayudará a entender el comportamiento de los errores y verificar que su tendencia es, como se espera, a la baja.

La ecuación de la recta para los mínimos cuadrados es:

$$y = A + Bx$$

donde B es la **pendiente** de la recta y es lo que nos indicará el comportamiento de la recta de ajuste. Para la obtención de los valores de A y B se utilizaron la formulas siguientes:

$$B = \frac{SCxy}{SCx} \qquad A = \bar{y} - B\bar{x}$$

de donde,

$$SCx = \sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n} \qquad SCxy = \sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

[4] Mendenhall, William trad. De la Fuente, Arturo Estadística Matemática con aplicaciones Iberoamérica, México, 1986, pp. 439.

reemplazando las valores para cada unode los casos, tenemos que las pendientes son:

sistema	error	pendiente
nómina	File is in use.	-0.256666807
	Subscript out of bounds.	-2.10849421
	Too many READs in effect.	-0.00790749
cuentas por pagar	Too many files open.	-6.2
	Variable not found.	-26.0857143
	Invalid subscript reference.	-3.15059289
materiales (MATISA)	File does not exist.	-6
	Database is not ordered.	-0.43296703
	Invalid subscript reference.	-0.52276215
contabilidad	File is in use.	-10
	Variable not found.	-7.26710454
	File access denied.	-5.94005602
materiales (IUSA)	Too many files open.	-15.9300699
	String memory variable area	-70.0714286
	Index file does not match db	-7.9
ventas	Bad drive specifier	-4.16883117
	CANCEL/SUSPEND is not	-28.569697
	Unable to create temporary	1.63760684

CAPITULO VI

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

1. Conclusiones

Al hacer un análisis de las pendientes de los diferentes errores encontramos que casi todas son negativas, indicandonos que efectivamente, el tiempo de respuesta a las fallas presenta una disminución en su tendencia. Por otro lado, si observamos las gráficas 5.8, 5.10, 5.12, 5.14, 5.16 y 5.18, los puntos (con excepción de la gráfica 5.12) se encuentran marcadamente cargados hacia el lado izquierdo, indicandonos que los errores que se cometieron están más alejados del presente.

Analicemos las pendientes de los sistemas de ventas y materiales de IUSA que fueron atendidos por un equipo de personas muy independientes de aquellos equipos de desarrollo y análisis (son sistemas comerciales de manufactura y administración). Ambos muestran una tendencia a la baja de sus tiempos de respuesta, excepto el último error que muestra una tendencia a la alta debido a que el problema aquí encontrado se debe principalmente a la capacidad del hardware que habia sido superada. El hecho de que su pendiente sea menor (mientras más se aleje el valor de la pendiente del cero, significa que decrece más rápido) que las pendientes de los sistemas atendidos por una sola persona se debe a que sus bases de conocimiento incluian conocimiento de tanto el personal del área como todo el generado por las firmas desarrolladoras de los sistemas. Es decir, su expertise aumentaba debido al conocimiento de los demás (aunque no los conociera) y sus tiempos de respuesta fueron mejorando mucho en el transcurso del tiempo.

En una primera impresión (esto debido a que nunca se midierón, y tampoco se obtuvieron los costos prorrateados, algunos de los recursos utilizados para solucionar problemas tales como

horas hombre, costos telefónicos, tiempo máquina, utilización de discos duros, etc.) el costo aportado por la utilización de recursos también bajo; aunque los costos por mantenimiento y administración de las bases de conocimiento (costo nuevos) hacen que de nuevo suba su porcentaje; este costo es despreciativo si los beneficios que reporta ayudan a la organización a mejorar sus procesos y la calidad de sus productos.

En conclusión, la **evaluación del impacto en tiempo y recursos que tiene en el área de sistemas de las organizaciones la tecnología de información utilizada como bases de conocimiento, aplicada en la recuperación de errores de sistemas de computo,** resulta en una clara disminución de los recursos asignados a esos proyectos, una satisfacción mayor de los clientes y un replanteamiento en el uso de las de las tecnologías de información aplicadas como bases de conocimiento para beneficio de toda la empresa; además en consecuencia, los incentivos hacia el área operativa de informática fueron incrementados de forma significativa, ya que las gerencias ahora cuentan con variables más tangibles a través de estos nuevos indicadores con las cuales medir su desempeño.

El uso de las bases de conocimiento resultó ser una experiencia muy agradable para los operadores y sobre todo, para los jefes de área, debido a que la información y tips sobre errores de codificación, problemas y fallas del compilador, etc. fueron compartidas de manera más **rápida** y **confiable**; también se logro que la información fluyera de forma natural entre la gente, aún en grandes distancias, y con ello un compartir de conocimientos más "sano" ya que en las cinco empresas se logró instalar la filosofía de **todos ganan** al compartir su conocimiento con los demás.

2. Recomendaciones y trabajos futuros

No es una labor fácil el aceptar los errores y mucho menos es reportar los tiempos que llegan a tardar en repararse y los recursos humanos empleados. El hacer que el personal lo tome

como una actividad que beneficiará a corto plazo a la empresa (y por supuesto a él mismo) es una tarea que es parte del tema de compartir el conocimiento (no hay culpables) y debe ser el punto angular en cualquier tipo de investigación que incluya la administración de conocimientos.

El desarrollo de este tipo de investigaciones trae implícitos varios factores que se deberán de tomar en cuenta al momento de plantearla a las distintas áreas de las organizaciones. Todo humano busca de alguna manera conservar su trabajo y es natural que evite, de alguna forma, participar en controles que muestren su desempeño, si es que la dirección o mando superior no le informa de manera adecuada los objetivos que se persiguen al establecerlos. En cambio, cuando se ha tenido una sesión informativa que contesta a las preguntas e inquietudes sobre el establecimiento de controles y se establece el por qué de ellos, la transición hacia su uso es más amigable.

Una investigación que puede resultar de interés es el realizar un análisis costo beneficio de hardware, software y humanware empleados para corregir los errores de los sistemas; de antemano podemos imaginar que es viable, pero ¿hasta qué punto?, ¿Cuánto es bueno invertir?, ¿Qué características deben tener las organizaciones que quieran incursionar en ella?, ¿Sería mejor manejar este tipo de proyectos por outsourcing?, etc.

La capitalización del conocimiento esta dando mejores resultados día a día. Cada vez es más común el escuchar en las organizaciones sobre manuales de procedimientos, manuales de operación y manuales de recuperación de errores. Todo lleva un mismo camino: el dejar registrada la historia de la organización para así no cometer los mismos errores que en el pasado y con ello ganar tiempo, recursos y un mejor producto.

ANEXO I

ISO 9000

GRUPO CRASA Y ASOCIADOS, S.C.

Excelencia a través de la mejora continua

Las 10 preguntas más frecuentes en torno a ISO 9000

¿Qué es ISO?

La Organización Internacional para la Normalización (ISO) es la entidad responsable para la normalización a escala mundial con una agrupación hasta la fecha de 91 países. La Dirección General de Normas (DGN) de la Secretaría de Comercio y Fomento Industrial (SECOFI) es la representante de ISO en México. ISO está formado por aproximadamente 180 comités técnicos, cada uno de los cuales es responsable de la normalización para cada área de especialidad desde, por ejemplo, asbestos hasta el zinc. El propósito de ISO es promover el desarrollo de la normalización para fomentar a nivel internacional el intercambio de bienes y servicios y para el desarrollo de la cooperación en actividades económicas, intelectuales, científicas y tecnológicas. El resultado del trabajo técnico dentro de ISO se publica en forma final como normas internacionales.

¿Quién desarrolló la serie ISO 9000?

El Comité Técnico 176 (ISO/TC 176) se formó en 1979 para armonizar la creciente actividad a nivel mundial en administración y aseguramiento de calidad. El Subcomité 1 se estableció para la normalización de términos, lo cual dio como resultado la norma ISO 8402 en 1986 y el Subcomité 2 emitió en 1987 las cinco normas que originalmente integraban a la serie ISO 9000. En 1994 se emitió la nueva revisión a estas normas y hasta la fecha la serie 9000 se compone de casi 20 normas, entre normas de guía y normas contractuales.

¿En qué consiste la serie ISO 9000?

ISO 9000 es una serie de normas que principalmente se dividen en normas de guía y normas contractuales. Las normas de guía sirven para aclarar algunos requisitos contenidos en las normas contractuales, que son aquellas normas que están sujetas a certificación y que se aplican especialmente cuando existe una relación entre dos partes, cliente y proveedor, mediados por un contrato. Las normas contractuales son 3:

ISO 9001:1994, la cual es aplicable a empresas cuyas actividades abarcan desde el diseño y desarrollo, pasando por fabricación, instalación y servicio. Este modelo es aplicable, por ejemplo, a las empresas de ingeniería que parten desde un proyecto. Si el producto o servicio que se ofrece requiere de un diseño para cumplir con requisitos establecidos por el cliente, la empresa tiene que apegarse al modelo de ISO 9001:1994.

ISO 9002:1994 es aplicable a empresas que parten de especificaciones ya establecidas. Ejemplos de empresas con este modelo serían subsidiarias de empresas automotrices o de electrodomésticos o empresas de supervisión, las cuales reciben especificaciones de sus matrices para fabricar el producto o prestar un servicio.

ISO 9003:1994, se refiere exclusivamente a inspección y pruebas finales. Ejemplos de empresas que pueden apegarse a este modelo son comercializadoras o distribuidoras, laboratorios de prueba o de control de calidad, fotocopiado, servicios de limpieza, etc.

¿Existen normas nacionales que equivalgan a ISO 9000?

Sí. México adoptó la serie ISO 9000 a fines de los años ochenta como Norma Oficial Mexicana como la serie NOM-CC. A raíz de la emisión de la Ley Federal de Metrología y Normalización en 1992, se cambió la nomenclatura a NMX o Norma Mexicana, la cual a diferencia de las NOM que son

obligatorias, son normas voluntarias. La serie NMX-CC es equivalente con la serie ISO 9000 de la NMX-CC-001 hasta la NMX-CC-008. El Comité Técnico de Normalización en Sistemas de Calidad (COTENNSISCAL) es el responsable de la elaboración y revisión de estas normas mexicanas equivalentes a la serie ISO 9000.

¿Dónde puedo adquirir estas normas?

La serie NMX-CC completa, se vende en el Instituto Mexicano del Petróleo, sede del Comité Técnico de Normalización en Sistemas de Calidad (COTENNSISCAL) , la Asociación Mexicana de Calidad o el Instituto Mexicano de Normalización y Certificación (IMNC).

Las normas ISO pueden solicitarse a través de bancos de normas ya establecidos en México, aunque la desventaja es su precio y la limitación propia a personas que no leen inglés. Prácticamente conviene adquirir la norma mexicana la cual es equivalente a las normas ISO.

Si mi empresa se certifica a la norma aplicable de la serie NMX-CC, ¿es reconocida su certificación en el ámbito mundial?

La respuesta es no. La NMX-CC es reconocida sólo en México, a pesar de su equivalencia con ISO 9000.

He escuchado que algunos organismos "internacionales" que ya operan en México, si ofrecen certificados reconocidos mundialmente.

Aunque es verdad que algunos organismos de certificación extranjeros si cuenten con el reconocimiento de sus certificados en varios países, hasta ahora no existe un certificado que sea válido a nivel mundial.

¿Existen organismos de certificación acreditados en México?

Hasta la fecha existen 2: Calidad Mexicana Certificada, A.C. (CALMECAC) y el Instituto Mexicano de Normalización y Certificación (IMNC), todos ellos ubicados en la Ciudad de México.

¿Qué tiempo lleva implantar ISO 9000 y que costos incvolucra?

El tiempo invertido en implantar ISO 9000 varía en función del tamaño de la empresa, la complejidad de sus procesos, que tan regulado está el producto que vende y obviamente, el compromiso de la Alta Dirección y del personal de la empresa.

Generalmente se habla de un tiempo mínimo de 6 meses, para una empresa micro, 8 meses para empresas pequeñas y medianas y hasta 2 años para empresas grandes.

Los costos involucrados son el tiempo del personal, asesoría externa y del organismo de certificación.

¿Y qué relación hay con Calidad Total, el Premio Nacional de Calidad, el Control Estadístico del Proceso, Reingeniería?

En realidad es erróneo pensar en ISO 9000 como un fin per se. Esto es tan solo una parte de la estructura sobre la cual se debe fundamentar la administración por calidad en una empresa como un inicio para creara una cultura de calidad y lograr el mejoramiento continuo. ISO 9000 es visto por muchos expertos como un buen inicio para un programa de calidad, porque representa la destilación de las mejores prácticas de administración de la calidad. La reingeniería y otras herramientas o modelos son igualmente útiles (el Premio Nacional de Calidad es una herramienta excelente de autoevaluación para procesos de calidad total) y pueden complementar a ISO 9000, aunque hay que tener la precaución al emplearlas y evitar su uso como si se tratara de recetas,

fórmulas mágicas o de éxito, así como contemplar el proceso ISO 9000 como estratégico. La ventaja de ISO 9000 es que ha sido un éxito dentro del campo de la normalización porque por primera vez existe una serie que representa el consenso mundial sobre las mejores prácticas administrativas para la calidad.

ANEXO II

SOFTWARE ASSURANCE STANDARD – NASA

INTRODUCTION

The NASA Software Assurance Standard (hereinafter referred to as the "Standard") specifies at a high level an overall NASA Software Assurance Program for software developed for and by NASA. It provides a consistent, uniform basis for specifying and defining the software assurance program to be applied by the developer of NASA software.

This Standard is at the highest level of software assurance standards developed by NASA. Further issuances of NASA software assurance standards will specify in more detail specific processes within the software assurance discipline.

The Standard will have been successfully applied if the:

- NASA software acquirer tailors it to meet the needs and requirements specific to a software development activity.
- Provider of software for NASA develops a software assurance program that meets the tailored requirements.
- Implemented software assurance program reduces the technical and programmatic risks associated with the delivery of software meeting NASA's technical, schedule, and budgetary needs.

The major sections of this standard are as follows:

- Section 1.0, Scope, Purpose, and Application, describes the function and type of activity and provides information on tailoring it to specific projects.
- Section 2.0, References, provides a listing of documents referenced and a glossary of terms used in the Standard.

- Section 3.0, Requirements, contains specific requirements for the software assurance program for providers of NASA software.
- Section 4.0, Quality Assurance Provisions, does not apply to this Standa assurance standard.
- Section 5.0, Packaging, does not apply to this Standard.
- Section 6.0, Additional Information, provides material that is general in nature, but which is not mandatory or contractually binding.

1.0 SCOPE, PURPOSE, AND APPLICATION

1.1 SCOPE

This Standard specifies the software assurance program for the provider of software. It also delineates the assurance activities for the provider and the assurance data that are to be furnished by the provider to the acquirer. In any software development effort, the provider is the entity or individual that actually designs, develops, and implements the software product, while the acquirer is the entity or individual who specifies the requirements and accepts the resulting products.

1.2 PURPOSE

This Standard specifies at a high level an overall software assurance program for software developed for and by NASA. Assurance includes the disciplines of Quality Assurance, Quality Engineering, Verification and Validation, Nonconformance Reporting and Corrective Action, Safety Assurance, and Security Assurance. The application of these disciplines during a software development life cycle is called Software Assurance. Subsequent lower-level standards will specify the specific processes within these disciplines.

1.3 APPLICATION

The NASA Software Assurance Standard and other NASA standards should be applied to all software developed for and by NASA, including the following:

1.3.1 Deliverable software.

1.3.2 Software included as part of deliverable hardware (including firmware).

1.3.3 Nondeliverable software (commercially available or user-developed) used for development, fabrication, manufacturing process control, testing, or acceptance of deliverable software or hardware (test and acceptance software; software design, test, and analysis tools; compilers, etc.).

1.3.4 Commercially available, reused, or government-furnished software (GFS) designated as a deliverable item or a part thereof.

1.4 TAILORING

This Standard shall be tailored by the acquirer (e.g., NASA program/project manager) in accordance with the classification of the software being developed or acquired. The classification of software is determined by the responsible NASA center or program office per NMI 2410.10A, "NASA Software Management, Assurance, and Engineering Policy." Tailoring of this Standard consists of the following:

1.4.1 Identifying requirements that are not applicable.

1.4.2 Adding requirements.

1.4.3 Providing quantifiable criteria for the requirements (how often, how many, quality criteria, etc.).

2.0 REFERENCES

2.1 REFERENCED DOCUMENTS

NMI 2410.7A Assuring the Security and Integrity of NASA Automated Information Resources, July 8, 1988.

NMI 2410.10A NASA Software Management, Assurance, and Engineering Policy, December 12, 1991.

NHB 1700.1 NASA Basic Safety Manual, Volume 1-B, Chapter 5.

NASA-STD-2100 NASA Software Documentation Standard, July 29, 1991.

NASA-GB-A201 Software Assurance Guidebook, September 1989, NASA.

NASA-GB-A301 Software Quality Assurance Audits Guidebook, December 1990, NASA.

NASA-GB-A302 Software Formal Inspections Guidebook, August 1993

2.2 GLOSSARY

Acceptance - Final approval and receipt of a product by its acquirer.

Acceptance Review - Phase transition review for the Acceptance and Delivery life cycle phase.

Acquirer - An organization that obtains a product or capability, such as a software system.

Architectural Design Review - See Preliminary Design Review.

Assurance - Those activities, regardless of the organization conducting the activities, that demonstrate the conformance of a product or process to a specified criteria (such as a functional requirement or a standard).

Assurance Plan - A document containing the technical and planning aspects of the assurance activities for a system/software development or acquisition project.

Audit - An activity to determine through investigation the adequacy of, and adherence to, established standards, procedures, instructions, specifications, or other applicable contractual and licensing requirements, and the effectiveness of implementation.

Baseline - A specification or product that has been reviewed formally and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

Configuration Management - Process of identifying and defining the baseline items in a system, controlling the release and change of these items throughout the system/software life cycle, and recording and reporting the status of baseline items and change requests.

Critical Design Review - Phase transition review for the Detailed Design life cycle phase.

Documentation - Any written or pictorial information annotating, describing, defining, specifying, reporting, or certifying activities, requirements, procedures, results, or products.

Firmware - Hardware that contains a computer program and data that cannot be changed in its user environment. The computer program and data contained in the firmware are classified as software; the circuitry containing the computer program and data are classified as hardware.

Formal Inspections - In-process technical reviews of a product of the software life cycle conducted for the purpose of finding and eliminating defects. Formal inspections may be applied to any product or partial product of the software development process, including requirements, design, and code.

Formal Reviews - See phase transition review.

Hardware - Physical equipment used in data processing, as opposed to computer programs, procedures, rules, and associated documentation.

Independent Verification and Validation (IV&V) - A process whereby the products of the software development life cycle phases are independently reviewed, verified, and validated by an organization that represents the acquirer of the software and is completely independent of the provider.

Integration - Process of combining software elements, hardware elements, or both into an overall system.

Life Cycle - Period of time that starts when a software product is conceived and ends when the software is no longer available for use. The traditional "waterfall" software life cycle has eight phases: Concept and Initiation; Requirements; Architectural Design; Detailed Design; Implementation; Integration and Test; Acceptance and Delivery; and Sustaining Engineering and Operations.

Metric - Quantitative measure of extent or degree to which software possesses and exhibits a certain characteristic, quality, property, or attribute.

Nonconformance - A deviation from specified standards, procedures, plans, requirements, or design.

Nonconformance Reporting and Corrective Action - The process used to identify, report, track, and correct nonconformances.

Phase - Period of time during the life cycle of a project in which a related set of software engineering activities is performed. Phases may overlap.

Phase Transition Review - Review at the end of a life cycle phase.

Preliminary Design Review - Phase transition review for the Preliminary (Architectural) Design Life Cycle Phase. Also, known as Architectural Design Review.

Program - NASA Headquarters organization and activities that funds and manages Center-level projects.

Project - Organization and associated activities that produce and/or provide a software system/product.

Provider - Organization that actually develops the software products to the requirements of the acquirer. May be a contractor or an in-house NASA entity.

Quality Assurance - Those assurance activities focused onconformance to standards and procedures.

Quality Engineering - Process of incorporating reliability, maintainability, and other quality factors into software products.

Requirements Analysis - Process of studying user needs to arrive at a specification of system or software requirements.

Requirements Allocation - Process of distributing requirements of a system to subordinate software and hardware elements.

Requirements Review - Phase transition review for the Requirements life cycle phase.

Risk - Combined effect of the likelihood of an unfavorable occurrence and the potential impact of that occurrence.

Risk Management - Process of assessing potential risks and reducing those risks within budget, schedule, and other constraints.

Software - Programs, procedures, rules, and associated documentation and data pertaining to the operation of a computer system. Includes programs and data contained in firmware.

Software Assurance - Planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures. It includes the disciplines of Quality Assurance, Quality Engineering, Verification and Validation, Nonconformance Reporting and Corrective Action, Safety Assurance, and Security Assurance and their application during a software life cycle.

Test Readiness Review - Phase transition review for the Integration and Test life cycle Phase.

Testing - Process of exercising or evaluating software by manual or automated means to demonstrate that it satisfies specified requirements or to identify differences between expected and actual results.

Verification and Validation - Verification is the process that assures that the software has "been built right"; that is, that each intermediate product during the life cycle meets its specific requirements. Validation is the process of evaluating software to assure that "the right product has been built"; that is, to assure that it meets its functional and performance requirements.

2.3 ABBREVIATIONS AND ACRONYMS

GFS - Government Furnished Software

IV&V - Independent Verification and Validation

NHB - NASA Handbook

NMI - NASA Management Instruction

NRCA - Nonconformance Reporting and Corrective Action

RFP - Request for Proposal

SEB - Source Evaluation Board

SOW - Statement of Work

SQA - Software Quality Assurance

SQE - Software Quality Engineering

V&V - Verification and Validation

3.0 REQUIREMENTS

3.1 GENERAL

A software assurance program shall be planned, documented, and implemented for software development activities. The software assurance program shall:

3.1.1 Ensure that assurance requirements are documented and satisfied throughout all phases of the life cycle.

3.1.2 Detect actual or potential conditions that could degrade quality, including deficiencies and system incompatibilities, and provide a process to ensure corrective action is taken and completed.

3.1.3 Assure timely and effective preventive action by identifying root causes of deficiencies and nonconformances.

3.2 SOFTWARE ASSURANCE

3.2.1 SOFTWARE ASSURANCE PLAN

For each development effort, the software assurance process to be applied shall be documented in a software assurance plan in accordance with NASA-DID-M400, contained in the NASA-STD-2100-91, "NASA Software Documentation Standard." The plan shall describe how the activities specified by this assurance standard will be implemented. The software assurance plan shall be reviewed and, if needed, updated at the end of each life cycle phase. The plan shall address, but is not limited to:

3.2.1.1 Descriptions of the procedures for each software assurance task including traceability to assurance program requirements.

3.2.1.2 Description of the role of the software assurance program in activities for continuous improvement such as:

3.2.1.2.1 A strategy that emphasizes prevention, not correction.

3.2.1.2.2 Improved use of tools and techniques.

3.2.1.2.3 Collection and evaluation of metric data.

3.2.1.2.4 Suggestions for improvements in assurance methods.

3.2.2 SOFTWARE ASSURANCE MANAGEMENT

The provider shall designate a software assurance manager who shall be responsible for directing and managing the software assurance program. The software assurance manager shall have a reporting channel to provider management that is independent of the provider's project management and software development function.

3.2.3 SOFTWARE ASSURANCE RECORDS

Records shall be prepared that contain the descriptions and results of all software assurance activities. Results, such as status reports and audit reports, shall include recommended preventative measures and corrective actions. These records shall be available to the acquirer.

3.2.4 SOFTWARE ASSURANCE STATUS REPORTING

Software Assurance status reports shall be prepared by the provider on its software assurance activities on a periodic basis, as specified by the acquirer. The report shall include:

3.2.4.1 Organization and key personnel changes.

3.2.4.2 Assurance accomplishments such as inspection and test activities, reviews, contractor/subcontractor surveys, etc.

3.2.4.3 Subcontractor assurance accomplishments, including items listed above, plus summaries of acceptance and certification reports.

3.2.4.4 Significant problems, their solutions, and remedial and preventive actions.

3.2.4.5 Significant trends in metric data.

3.2.4.6 Recommendations and lessons learned.

3.2.5 SOFTWARE ASSURANCE PLAN CHANGE PROCEDURES

Any proposed deviations from or modifications to the baselined software assurance plan shall be submitted to the acquirer as a formal change request. Proposed changes shall be accompanied by a risk analysis performed to identify the potential impact of the change.

3.2.6 SOFTWARE ASSURANCE APPROVAL AUTHORITY

The software assurance manager shall have approval authority on the establishment and composition of all software baselines and any changes to the baselines.

3.3 SOFTWARE ASSURANCE FUNCTIONS

3.3.1 SOFTWARE QUALITY ASSURANCE

Software Quality Assurance (SQA) is concerned with the evaluation of the quality of, and adherence to, software-related standards and procedures. SQA activities shall be performed during each phase of the software life cycle.

3.3.1.1 The SQA process shall ensure that:

3.3.1.1.1 Standards and procedures for management, engineering, and assurance activities are specified.

3.3.1.1.2 Management, engineering, and assurance adhere to specified standards and procedures.

3.3.1.1.3 All documentation and report formats and content descriptions are defined.

3.3.1.1.4 All plans (configuration management, risk management, assurance plan, management plan, etc.) are completed and implemented according to specified standards and procedures.

3.3.1.1.5 The configuration management process functions according to approved procedures and that all baselined items are maintained under Configuration Management.

3.3.1.1.6 Baseline control, configuration identification, configuration control, configuration status accounting, and configuration authentication activities are carried out.

3.3.1.2 The SQA process shall include the following activities:

3.3.1.2.1 Evaluation of specified standards and procedures.

3.3.1.2.2 Audits of all management, engineering, and assurance processes, for example, Configuration Management.

3.3.1.2.3 Reviews of all project documentation including reports, schedules, and records.

3.3.1.2.4 Monitoring of formal inspections and formal reviews.

3.3.1.2.5 Monitoring/witnessing of formal and acceptance-level software testing.

3.3.2 SOFTWARE QUALITY ENGINEERING

The Software Quality Engineering (SQE) process is concerned with incorporating reliability, maintainability, usability, and similar requirements into the products produced at each phase of the life cycle.

3.3.2.1 The SQE process shall ensure that:

3.3.2.1.1 All quality requirements are defined in a manner that is measurable or otherwise verifiable.

3.3.2.1.2 Quality requirements are considered during design of the software.

3.3.2.1.3 The software is tested/measured to verify compliance with quality requirements.

3.3.2.2 The SQE process shall include the following activities:

3.3.2.2.1 Analysis, identification, and detailed definition of all quality requirements.

3.3.2.2.2 Quality engineering evaluations of standards, design, and code.

3.3.2.2.3 Collection and analysis of metric data pertaining to quality requirements.

3.3.3 SOFTWARE VERIFICATION AND VALIDATION

Software Verification and Validation (V&V) is concerned with ensuring that software being developed or maintained satisfies functional and other requirements and that each phase of the development process yields the right products.

3.3.3.1 V&V activities shall be performed during each phase of the software life cycle and shall include the following:

3.3.3.1.1 Analysis of system and software requirements allocation, verifiability, testability, completeness, and consistency (including analysis of test requirements).

3.3.3.1.2 Design and code analysis including design completeness and correctness.

3.3.3.1.3 Interface analysis (requirements and design levels).

3.3.3.1.4 Formal Inspections.

3.3.3.1.5 Formal Reviews (phase transition reviews).

3.3.3.1.6 Test planning, performance, and reporting.

3.3.3.2 If Independent Verification and Validation (IV&V) is specified, the IV&V activities shall be performed by an organization which reports directly to and is funded directly by the acquirer, and that is other than and independent of, the software provider. The software provider shall cooperate with the IV&V provider and shall facilitate access to all software and assurance products.

3.3.4 NONCONFORMANCE REPORTING AND CORRECTIVE ACTION

Software Nonconformance Reporting and Corrective Action (NRCA) is concerned with reporting, analyzing, and correcting nonconformances and collecting information from which reports on the overall status of nonconformances can be made. NRCA activities shall be performed during each phase of the software life cycle.

3.3.4.1 The NRCA process shall contain, but not be limited to, the following:

3.3.4.1.1 Nonconformance detection and reporting procedures.

3.3.4.1.2 Nonconformance tracking and management procedures.

3.3.4.1.3 Nonconformance impact assessment and corrective action procedures.

3.3.4.1.4 Interfaces to the Configuration Management process.

3.3.5 SOFTWARE SAFETY ASSURANCE

Software Safety Assurance is concerned with the satisfaction of system safety requirements that are allocated to the software, and the identification and verification of adequate safety controls and inhibits that are to be implemented in software.

3.3.5.1 The software safety program shall ensure that:

3.3.5.1.1 Safety-related deficiencies in specifications and design are identified and corrected.

3.3.5.1.2 Software design incorporates positive measures to enhance the safety of the system.

3.3.5.1.3 Software safety is included as an agenda item for formal reviews.

3.3.5.2 The software safety process shall include the following activities:

3.3.5.2.1 Determining the safety criticality for each software component.

3.3.5.2.2 Analyzing consistency, completeness, correctness, and testability of safety requirements.

3.3.5.2.3 Analysis of design and code to ensure that they correctly implement safety-critical requirements.

3.3.5.2.4 Analysis of changes for safety impact.

3.3.5.3 Software safety tasks assigned shall be in compliance with the intent of and the system-related activities required by NHB 1700.1, "NASA Basic Safety Manual." Whenever possible, software safety tasks shall be coordinated with the overall software assurance functions to eliminate duplication of effort.

3.3.6 SOFTWARE SECURITY ASSURANCE

Software Security Assurance is concerned with assuring that security requirements are planned, documented, and implemented during all phases of the software acquisition life cycle. Software security tasks and activities shall include the addressing of security concerns during reviews, analyses, inspections, testing, and audits. Software security tasks shall be in compliance with NMI 2410.7A, "Assuring Security and Integrity of NASA Automated Information Resources."

3.4 TRAINING

Personnel developing and implementing the software assurance process shall be trained and/or experienced in software assurance. Software assurance training shall be obtained and/or originated and maintained as necessary for management, engineering, and assurance personnel. Records shall be maintained and readily available for review of the training, testing, and certification/recertification status of personnel.

3.5 SUBCONTRACTOR CONTROLS

The provider shall be responsible for the adequacy and quality of all software, associated documentation, and services procured through subcontracted efforts. The provider shall flow down the requirements of this Standard to any subtier provider of software. Provider Software Assurance staff shall ensure acceptable subcontractor performance by:

3.5.1 Developing software assurance requirements for subcontractors.

3.5.2 Participating in the selection of subcontract sources.

3.5.3 Performing/participating in periodic surveys of each subcontractor's facility and software assurance processes to determine their capability of satisfying software requirements.

3.5.4 Reviewing procurement documentation issued at subcontractor facilities, prior to release, for adequacy of software assurance requirements.

3.5.5 Assuring the existence or development of a complete set of software requirements for subcontractor-developed software.

3.5.6 Examining and auditing subcontractor software development and assurance processes and procedures to determine compliance with requirements, standards, and procedures.

3.5.7 Evaluating subcontractor-developed software products for compliance with requirements and standards prior to acceptance.

4.0 QUALITY ASSURANCE PROVISIONS

This section is not applicable to this Standard.

5.0 PACKAGING

This section is not applicable to this Standard.

6.0 ADDITIONAL INFORMATION

This section contains additional information pertaining to this Standard. The material provided is intended for informative purposes only and should not be considered additional requirements. Further information on Software Assurance can be found in NASA Guidebooks such as NASA-GB-A201, "Software Assurance," NASA-GB-A301, "Software Assurance Audits", and NASA-GB-A302, "Software Formal Inspections."

The following paragraphs explain the acquirer's role in software assurance. Paragraphs 6.1 and 6.2 provide background information on some key concepts. Paragraph 6.3 describes the software acquisition process and the role of the acquirer's assurance efforts in that process.

6.1 RELATIONSHIP BETWEEN ACQUIRER AND PROVIDER

This Standard defines two types of organizations, acquirers and providers. The acquirer is the organization that is purchasing the software, product, or associated services. Usually, the acquirer is NASA or an organization within the Agency. The provider is the organization that is delivering that product or service. This can be a contractor, a separate NASA organization, or, in some cases,

the acquirer and the provider are the same organization. Providers may provide software or services such as IV&V, consulting, etc. A provider is ultimately responsible for providing a product or service to the acquirer according to the acquirer's requirements. The acquirer is ultimately responsible for the quality and effectiveness of all products.

The material in Section 3 of this Standard can be used by both an acquirer and a provider. Both organizations will perform the same types of activities and manage their assurance programs in a similar manner.

6.2 RELATIONSHIP BETWEEN PLANS, STANDARDS, AND PROCEDURES

Software plans (i.e., management plans, assurance plans, risk management plans) are prepared in conformance with specified documentation standards. The purpose of these plans is to document/specify the conduct of all activities. The plans will specify standards and procedures to be used by both the acquirer and the provider. The plans, once reviewed and approved by the designated authority, are used to determine what to assure and how to perform the assurance.

Standards and procedures establish the methods by which software will be managed, engineered, and assured, and are the criteria against which the products and processes will be measured. Standards and procedures, once selected and approved, must be tailored to the needs of each project before they are imposed. Tailoring of standards and procedures is the responsibility of the organization imposing them.

6.3 ACQUIRER'S ASSURANCE

The majority of software systems used at NASA are developed with NASA as the acquirer and a contractor as a provider. A typical scenario of how the acquirer should assure the project is given in the following paragraphs. Modifications to this scenario are needed if NASA also is a provider.

6.3.1 PRE-AWARD

Before awarding a contract for the project, the acquirer willcreate its requirements, management plan, assurance plan, and Request for Proposal (RFP) which includes a Statement of Work (SOW). These documents will be produced according to specified standards and/or procedures.

The acquirer should ensure that these standards and procedures are specified, tailored, and implemented by the acquirer and that the acquirer's requirements, management plan, and RFP are complete and adequate. The acquirers should particularly check the assurance requirements in the RFP. Specifically, all assurance requirements should be determined and related standards and procedures to be imposed on the provider should be selected, tailored, and included in the RFP.

The primary pre-award assurance activities carried out are reviews of the acquirer's requirements, management plan, and RFP. Assurance personnel should participate in all review and Source Evaluation Board (SEB) activities. The assurance manager should sign off on the acquirer's requirements, management plan, and the RFP.

Additionally, the acquirer's preliminary assurance plan should be completed. This plan should cover, as a minimum, all activities that will be performed before the awarding of the contract and all other activities not dependent on specific provider responses to the RFP.

6.3.2 POST-RFP, PRE-AWARD

Once proposals have been received from potential contractors, the acquirer evaluates the proposals in preparation for contract award. Pre-award surveys of prospective providers may be performed to fully assess their capabilities. The contract is then awarded.

Assurance personnel should participate in proposal evaluations, pre-award surveys, and review and approval of the contract. Assurance personnel should pay particular attention to prospective providers' assurance capabilities during evaluations and surveys.

6.3.3 POST-AWARD, PRE-DEVELOPMENT

Once the contract is awarded, the acquirer's assurance plan should be completed. Many details of an assurance plan cannot be completed until the provider is determined and the formal contract fixed. This is due to many reasons, including:

6.3.3.1 Standards and procedures to be used by the provider may not be finalized until the contract is final.

6.3.3.2 Assurance requirements may not be finalized until the contract is awarded.

The provider will prepare their preliminary management and assurance plans for acquirer review and approval.

6.3.4 DEVELOPMENT

During development, the software requirements are defined, the designs are finalized, the software is coded, and the software system is integrated and tested. The acquirer should beprimarily concerned with:

6.3.4.1 Reviewing the provider's assurance process status reports regularly to ensure that the provider is performing all specified activities.

6.3.4.2 Participating in all end-of-phase reviews.

6.3.4.3 Participating on the Change Control Board.

6.3.4.4 Reviewing proposed changes to the acquirer's assurance plan.

6.3.4.5 Reviewing the acceptance test plan.

Additionally, the acquirer's assurance process should contain some activities to ensure that the provider is adhering to approved plans and procedures and that these plans and procedures are effectively fulfilling their purpose. These activities may include audits, reviews, analyses, etc.

The acquirer's assurance process must also include oversight andevaluation of the acquirer's management and engineering processes. Specifically, reviews, audits, and evaluations should be performed to ensure adherence to and effectiveness of approved plans and procedures.

6.3.5 ACCEPTANCE

During acceptance, the acquirer verifies that the software and all related products (code, documentation, etc.) are complete and that they meet all of the specified requirements. The acquirer will:

6.3.5.1 Review a functional demonstration of the software conducted by the provider and/or perform acceptance testing of the software to assure that it meets its requirements.

6.3.5.2 Review all acceptance and delivery documentation for completeness and accuracy.

The acquirer must ensure that any acquirer facilities (buildings, hardware, etc.) are prepared to receive and implement use of the software.

ANEXO III

NOTES ON AS 274 "LEAST-SQUARES ROUTINES TO SUPPLEMENT THOSE OF GENTLEMAN"

by Alan Miller.

This algorithm provides a set of high accuracy least squares routines which expand upon those provided by Gentleman in AS 75. In particular, it includes facilities for (weighted) least-squares for a subset of the variables, for changing the order of the variables, for testing for singularities, and for calculating an estimated covariance matrix for the regression coefficients.

The algorithm is NOT consistent with those which have been published in the same journal by Clarke (AS 163), Stirling (AS 164) or Smith (AS 268), in that the orthogonal reduction is stored in a different way. It is unfortunate that these authors have not used the same format as Gentleman.

The basic algorithm is the same as Gentleman's. As each new case is added, the orthogonal reduction is updated. In traditional least-squares notation, the algorithm goes straight from the X-matrix (the 'design' matrix) to the Cholesky factorization WITHOUT the intermediate step of forming a sum of squares and products matrix. Thus it avoids squaring the condition number. To be pedantic, it is actually the Banachiewicz factorization which is used.
We use:

$$Q\,X = \mathrm{sqrt}(D)\,R$$

where Q is an orthonormal matrix such that Q'Q = I, D is a diagonal matrix (sqrt(D) is my crude way of indicating a diagonal matrix with elements on the diagonal which are the square roots of the elements stored in array D in the routines), and R is an upper triangular matrix with 1's on the

diagonal. The array R is stored in a 1-dimensional array with elements stored by rows as shown below (the 1's on the diagonal are NOT stored).

Storage of R: (1) 1 2 3 4 5

(1) 6 7 8 9

(1) 10 11 12

(1) 13 14

(1) 15

(1)

N.B. The matrix Q is a product of planar-rotation matrices (Jacobi/Givens rotations); it is not calculated or stored. N.B. If you want to fit a constant (intercept) in your model, you can do it by putting the first element of each row of X equal to 1.0. N.B. You may like to dimension XROW as XROW(0:K) in your calling program, where K is the number of variables (other than the constant). The parameter NP passed to routine INCLUD (and others) then takes the value K+1.

To perform the calculation of regression coefficients (and nothing else), you need to call the following routines in the order given:

1. clear - initializes arrays
2. includ - call once for each case to update the orthogonal reduction
3. tolset - calculates tolerances for each variable
4. sing (optional) - tests for singularities
5. regcf - calculates regression coefficients for the first NREQ variables

You can then add extra data and recalculate the regression coefficients.

Other routines (all require that 1 and 2 above have been used first).

ss calculates array RSS such that RSS(I) = the residual sum of squares with the first I variables in the model

cov calculates the estimated covariance matrix of the regression coefficients (the user must input a value for the residual variance VAR, usually the residual sum of squares SSERR divided by NOBS - NP for all variables, or RSS(I) divided by NOBS - I for the first I variables).

pcorr calculates partial correlations with the first IN variables in the model. If IN = 1 and the first variable is identically equal to 1 then these are the usual full correlations. If IN = 0 it gives the cross products (not centered) divided by the square root of the product of the second moments (about zero) of the two variables.

vmove moves the variable in position FROM to position TO.

reordr calls vmove repeatedly to re-order the variables from the current order in integer array VORDER to that in the integer array LIST.

hdiag calculates the diagonal elements of the `hat' matrix used in various diagnostic statistics.

For vmove and reordr, the user must first set up an integer array VORDER which assigns a unique integer value with each variable. You may like to associate the value 0 with the constant in the model.

For cov, vmove, reordr and hdiag you must call tolset first.

*** Warning *** Routine INCLUD (from Gentleman's AS 75) overwrites the elements in array XROW.

Alan Miller

25 November 1991

Notes on handling singularities

If there is a singularity amongst your X-variables and routine SING is not used, you will get wrong answers from most routines. The orthogonal reduction can be written as:

$$X = Q' \, \mathrm{sqrt}(D) \, R$$

If we denote the columns of X as X1, X2, ..., Xk, and the columns of Q' as q1, q2, ..., qk, then we have:

$$x1 = r(11).q1$$

$$x2 = r(12).q1 + r(22).q2$$

$$x3 = r(13).q1 + r(23).q2 + r(33).q3$$

etc., where r(ij) = the (i,j)-element of R multiplied by the square root of the i-th diagonal element of D. That is x1 is equal to the first orthogonal direction, q1, multiplied by a scaling factor. x2 = a projection of length r(12) in the first direction, q1, plus a projection of length r(22) in a new direction, q2, which is orthogonal to q1, etc. If the X-directions are all orthogonal then all of the r(ij)'s for off-diagonal elements of R will be zero.

Let us suppose that there is a singularity amongst the X-variables, that is that one of the X-variables is exactly linearly related to some of the others. For simplicity, let us suppose that x3 = a.x1 + b.x2. The direction q3 is formed from x3 by subtracting its projections in directions q1 and q2 and then scaling so that it has length equal to 1. When the projections in directions q1 and q2 are subtracted in this case, there should be nothing left. In practice, there will be almost always be small rounding errors. Direction q3 will be formed from these rounding errors. Thus we will have a rogue direction in the columns of Q'. The projections of the dependant variable on this direction can be large. It is like correlating the Y-variable on a variable formed by using random numbers or

a column of numbers from the phone directory. Routine SING sets the projections on this direction equal to zero.

In the case just mentioned, a full rank X could be obtained by removing any one of the variables x1, x2 or x3. In AS274, the equivalent to removing a variable is to move it to the last position and then to only use the first (k-1) variables in subsequent calculations. You still tell routines such as REGCF that there are NP variables, but you set NREQ = NP - 1.

If you just want properties of a subset of variables, then you use either VMOVE or REORDR to re-order the variables so that the first ones are those which you are interested in, and you set NREQ equal to the number of those variables. In most cases, the constant (intercept) will be included in the model and so will be included in the count of variables, NREQ (N-required).

Features NOT included in AS274

There is no facility for removing or adding variables (columns). This could be done but adding variables requires the storage of the Q-matrix, or at least of the rotations used to form it.

There is no facility for two or more dependant variables, though this is fairly easy to program. The extra dependant variables can be treated as X-variables. Thus if we have dependant variables Y1, Y2 and Y3, then Y2 and Y3 can be added at the end of the list of X-variables. When looking at the properties of Y1 related to the X-variables, use only the first (NP-2) columns by setting NREQ = NP-2. When you want to look at Y2, just copy column (NP-1) of RBAR into THETAB - this requires a little thought to work out just where those elements are stored. You may want to first copy the original THETAB, or to copy the column of RBAR into a different array, perhaps THETAB2 say. The residual sum of squares, SSERR, for variable Y2 is D(NP-1).

Y3 is a little more difficult. You must first change the order of the variables so that Y3 is before Y2, otherwise you will be regressing Y3 on Y2, and you probably don't want to do that.

The easy way of thinking about the above is by considering each column of R (after scaling by the square roots of the diagonal elements of D) as the projections of the corresponding variable on all of those which have preceded it. The Y-variable is just another variable being projected upon a set of orthogonal directions formed by the variables which came before it. The vector THETAB is stored separately just for fast access as most users will have only one Y-variable; it could have been incorporated as the last column of RBAR.

Alan Miller

2 May 1993

http://www.hensa.ac.uk/ftp/mirrors/statlib/apstat/274

12.12.1998.

BIBLIOGRAFIA

- Beizer, Boris. Software System Testing and Quality Assurance, Van Nostrand Reinhold Co., New York, 1984.
- Boehm, B.W. Software Engenniering, The Institute of Electrical and Electronics Engineers, Inc., New York, 1979.
- Denton, D.K. The Learning Organization Involves the entire work force, Quality Progress, Dec. 1992.
- Deutsch, Michael; Willis, Ronald. Software Quality Engineering, Prentice-Hall, New Jersey, 1998.
- Dertouzoz, Michael, Qué será, Planeta, 1997.
- Flood, A.L., The Learning Organization, The worklife report, 1993.
- GRUPO CRASA Y ASOCIADOS, S.C. FAQ ISO 9000 10.08.1998.
- Guns, Bob. Aprendizaje Organizacional: Como Ganar y Mantener la Competitividad, Prentice-Hall, México, 1996.
- Hetzel, W.C. Program Test Methods, Prentic-Hall, New Jersey, 1972.
- IEEE Standards Board,IEEE Guide for Software Verification and Validation Plans, The Institute of Electrical and Electronics Engineers, Inc., New York, 1993.
- Manna, Zohar; Waldinger, Richard. The Logic of Computer Programming. IEEE Transactions on Software Engineering SE-4, 1978.
- Marquardt, Michael J. Building the Learning Organization, McGraw-Hill, 1995.
- Mendenhall, William trad. De la Fuente, Arturo Estadística Matemática con aplicaciones Iberoamérica, México, 1986.
- Myers, G.J. The Art of Software Testing, John Wiley and Sons, New York, 1979.
- NASA Standards Board SOFTWARE ASSURANCE STANDARD 05.05.1998.
- Stata, Ray, Organizational Learning, Sloan maagement review, Nov.-Dec. 1992.

- Senge, Peter. La Quinta Disciplina: Cómo Impulsar el Aprendizaje en la Organizacion Inteligente, Vergara, Argentina, 1990.
- Senge, Peter, La Quinta Disciplina, Vergara, Argentina, 1989.
- Bohn, Kathy, "Converting Data for Warehouses", URL: http://www.dbmsmag.com/706d15.htm, 06.08.1998.
- Fisher, Lawrence. Along the Infobahn. URL: http://www.strategy-business.com/ 04.07.1998.
- LaLiberte, Daniel About Hypernews http://www.hypernews.org/, 10.11.1998.
- Lotus Development Corporation, Información de productos Lotus, URL: http://www.lotus.com/, 30.04.1998.
- Miller, Alan NOTES ON AS 274 "LEAST-SQUARES ROUTINES TO SUPPLEMENT THOSE OF GENTLEMAN" http://www.hensa.ac.uk/ftp/mirrors/statlib/apstat/274 12.12.1998.
- Zack, Michael; Serino, M. Knowledge Management and Collaboration Technologies, Lotus Corporation, URL: http://www.lotus.com/services/institute.nfs/, 30.04.1998.