

## C A P I T U L O XII

### EL MANTENIMIENTO DEL SOFTWARE :

El mantenimiento ha sido caracterizado como un Lurte. Esperando que lo que sea inmediatamente visible sea todo lo que es. En realidad, se sabe que una enorme masa de --- problemas potenciales y los costos están debajo de la superfi--- cie. El mantenimiento del existente Software se estima sobre -- el 60% de todo el esfuerzo gastado en la organización de desa--- rrollo. El porcentaje continúa a incrementar como más Software es desarrollado. En el horizonte se puede ver la limitación --- del mantenimiento de la organización de desarrollo Software que no pueda producir nuevo Software porque está gastando todos los recursos disponibles en mantener el Software viejo.

Un lector no iniciado puede preguntar porque se requiere mucho mantenimiento y porque se gasta tanto esfuerzo. Rochkind provee una contestación parcial:

"Los programas de computadora siempre están cambiando. Hay errores que arreglar, engrandecimientos que agre--- gar, y optimizaciones que hacer. No hay solo la versión de este año para cambiar, pero también la versión del año pasado y la - versión del año que entra. Aparte a los problemas que sus solu--- ciones requieren los cambios en primer lugar, y los cambios mis--- mos crean problemas adicionales".

A través de este libro se ha discutido una metodología de Ingeniería Software. Su meta primaria es reducir el esfuerzo gastado en el mantenimiento. En este capítulo, la fase final del ciclo de vida del Software, el mantenimiento Software.

### UNA DEFINICION DEL MANTENIMIENTO DEL SOFTWARE :

Podemos definir el mantenimiento describiendo cuatro actividades que se emprenden después que el programa sea liberado para su utilización.

La primera actividad de mantenimiento ocurre porque no es razonable asumir que las pruebas del Software descubrirán errores latentes en un sistema grande. Durante la utilización de cualquier gran sistema, errores ocurrirán y serán reportados al desarrollador. El proceso que incluye el diagnóstico y corrección de uno o más errores se llama mantenimiento correctivo.

La segunda actividad que contribuye a la definición del mantenimiento ocurre por el cambio rápido que se encuentra en cada aspecto de la computación. Nuevas generaciones de Hardware se anuncian en ciclos de 36 meses, nuevos sistemas aparecen regularmente, equipo periférico y otros elementos del sistema son frecuentemente modificados.

La vida útil de la aplicación del Software puede sobre pasar los diez años, sobreviviendo el ambiente del sistema por el cuál fue originalmente desarrollado. Entonces, el mantenimiento adaptivo es la actividad que modifica el Software para que propiamente sea interfase del ambiente cambiante, es común

y necesario.

La tercera actividad que puede ser aplicada a la definición del mantenimiento ocurre cuando el paquete Software es exitoso. Como se utiliza el Software, se recomiendan nuevas capacidades, modificaciones a funciones existentes y engrandecimientos son recibidos por los usuarios. Para satisfacer las peticiones en esta categoría. Se ejecuta el mantenimiento perfecto. Esta actividad estima la mayoría del esfuerzo gastado en el mantenimiento del Software.

La cuarta actividad del mantenimiento ocurre cuando el Software es cambiado para mejorar el mantenimiento futuro ó confiabilidad o para proveer una mejor base para futuros engrandecimientos ó mejoramientos. A veces llamada mantenimiento preventivo, ésta actividad es todavía relativamente rara en el mundo del Software.

#### ✓ CARACTERISTICAS DEL MANTENIMIENTO :

El mantenimiento Software, hasta recientemente ha sido la fase negada en el ciclo de vida del Software. La literatura sobre mantenimiento contiene pocas entradas cuando es comparado con las fases de planeación y desarrollo. Poca investigación o datos de producción han sido recolectados sobre este tema y pocos métodos técnicos han sido propuestos.

Para entender las características del mantenimiento del Software, consideremos el tópicó de tres puntos de vista:

1. Las actividades requeridas para efectuar la fase de mantenimiento y el impacto del acceso de la Ingeniería de Software (a la falta de) sobre la eficiencia de tales actividades.
2. Los costos asociados con la fase de mantenimiento.
3. Los problemas que frecuentemente se encuentran cuando se efectúa el mantenimiento.

El costo del mantenimiento ha incrementado durante los pasados 20 años. Aunque los promedios industriales son difíciles de decir y están abiertos a una interpretación amplia, la organización típica de desarrollo Software gasta entre el 40 y 60% de las entradas en mantenimiento.

El costo del mantenimiento es muy obvio. -- Pero, costos menos tangibles puede ser la causa para mayor preocupación. Un costo intangible del mantenimiento del Software es la oportunidad que se ha pospuesto o perdido porque los recursos disponibles se deben canalizar a las tareas de mantenimiento. Otros costos intangibles incluyen:

- \* Insatisfacción del cliente cuando los pedidos legítimos para reparación ó modificación no pueden ser direccionados a tiempo.

- \* La reducción de la calidad general del Software como un resultado de los cambios que introducir errores -

latentes en el mantenimiento del Software.

- \* Solventamiento causado durante los esfuerzos de desarrollo cuando el staff deben de trabajar en tareas de mantenimiento.

El costo final del mantenimiento del Software es un decremento dramático en la productividad (medido en líneas de código (LOC)/persona-mes ó \$/LOC) que se encuentran cuando es -- iniciado el mantenimiento en programas viejos. Las reducciones - de productividad de 40 a 1 han sido reportadas. Esto es, un es - fuerzo de desarrollo que costó \$25.00 por LOC para desarrollar, puede costar \$1000.00 por LOC de mantención.

El esfuerzo gastado en mantenimiento puede ser - dividido en actividades productivas (análisis, evaluación, dise - ño de modificaciones y codificación) y otras actividades como - tratar de entender que hace el código, tratar de interpretar las estructuras de datos, características de las interfases y lími - tes de ejecución. La siguiente expresión provee un modelo para - el esfuerzo de mantenimiento.

$$M = p + K \exp(c-d)$$

donde M es el esfuerzo total gastado en mantenimiento, p es el - esfuerzo de productividad, K es una constante empírica, c es la medida de complejidad que se puede atribuir a la falta de estruc - turación del diseño y documentación, y d es la medida del grado de familiaridad con el Software.

El modelo indica que el esfuerzo y el costo pueden incrementar exponencialmente, si la metodología de desarrollo del Software es pobre (falta de Ingeniería Software) fue utilizada, y la persona o grupo que utilizó la metodología no está disponible para la ejecución del mantenimiento.

### MANTENIBILIDAD :

Las características descritas anteriormente son afectadas por la mantenibilidad del Software. La mantenibilidad puede ser definido cualitativamente como la facilidad, con la cuál, el Software es entendido, corregido, adaptado y/o mejorado. Como se ha enfatizado a través de este libro, la mantenibilidad es la meta primaria que guía los pasos de la metodología de la Ingeniería Software.

La mantenibilidad del Software es afectada por muchos factores. Descuido inadvertido en el diseño, codificación y pruebas tienen un impacto negativo en la habilidad de mantener el Software puede tener un impacto negativo similar, aún cuando los pasos técnicos han sido conducidos con cuidado.

Además de los factores que se pueden asociar con la metodología del desarrollo, Kopetz define un número de factores que están relacionados al ambiente de desarrollo:

- \* La disponibilidad de un staff calificado
- \* La estructura del sistema entendible
- \* La facilidad del manejo del sistema

- \* La utilización de lenguajes de programación estandarizado.
- \* La utilización de sistemas operativos estandarizados
- \* La estandarización de la estructura de documentación
- \* Disponibilidad de casos prueba
- \* Facilidades del debugging
- \* Disponibilidad de la computadora para conducir el mantenimiento.

Posiblemente, el factor más importante que afecta a la mantenibilidad es planear para la mantenibilidad. Si el Software es visto como un elemento del sistema que inevitablemente sufrirá cambios, los cambios que el Software mantenible que se producirán se incrementarán substancialmente.

La mantenibilidad del Software, como la calidad ó confiabilidad, es un término difícil de medir. Podemos ~~---~~ amillar indirectamente la mantenibilidad considerando los atributos de la actividad de mantenimiento que podemos medir. Gilb provee un número de medidas de mantenibilidad que se relacionan con el esfuerzo gastado durante el mantenimiento :

1. Tiempo de reconocimiento del problema
2. Tiempo de retardo administrativo
3. Tiempo de colección de herramientas de mantenimiento
4. Tiempo de análisis del problema
5. Tiempo de cambio de especificación
6. Tiempo activo de corrección ó modificación
7. Tiempo de pruebas locales

8. Tiempo de pruebas globales
9. Tiempo de revisión del mantenimiento
10. Tiempo total de recuperación

Cada una de éstas medidas pueden ser grabadas - - sin gran dificultad. Tales datos pueden proveer al administrador con una indicación de la eficiencia de técnicas y herramientas -- nuevas.

En cada nivel del proceso de revisión, la mante - nibilidad se ha considerado. Durante la revisión de requerimien - tos (Capítulo IV), se notaron las áreas de mejoramiento futuro y revisiones potenciales, los puntos de portabilidad del Software - se discutieron, y se consideraron las interfases del sistema que puedan tener un impacto en el mantenimiento. Durante las revisio - nes informales y formales (Capítulo V), la estructura y procedi - miento son evaluadas para la facilidad de modificación, modulari - dad e independencia funcional. Las revisiones del código (Capítulo X) enfatizan el estilo y la documentación interno, dos factores -- que tienen influencia con la mantenibilidad. Finalmente, cada paso de prueba (Capítulo XI) puede proveer sugerencias de partes del -- programa que pueda requerir mantenimiento preventivo antes de que sea formalmente liberado.

Las revisiones de mantenibilidad son conducidas - repetidamente como cada paso en el proceso de Ingeniería Software es completada. La revisión de mantenimiento más formal ocurre en - la conclusión de las pruebas y es llamada la revisión de configura - ción. Discutido en el Capítulo XI, la revisión de la configura - ción asegura que todos los elementos de la configuración del Soft-

ware estén completos, entendibles y archivados para un control de modificación.

# B I B L I O G R A F I A

## CAPITULO I.

A. OSBORNE, RUNNING WILD, THE NEXT INDUSTRIAL  
REVOLUTION.

Mc GRAW - HILL

C. BELL, J. MUDGE Y J. Mc NAMARA, COMPUTER INGINEE  
RING.

DIGITAL PRESS, DIGITAL EQUIPMENT Co.

T. BARTEE, DIGITAL COMPUTER FUNDAMENTALS

Mc GRAW - HILL.

## CAPITULO II.

J.C. WETHERBE, SYSTEMS ANALISIS FOR COMPUTER  
BASED INFORMATION SYSTEMS.

WEST PUBLISHING.

T.K. ORR, STRUCTURED SYSTEMS DEVELOPMENT  
YOURDON PRESS.

T. DE MARCO, STRUCTURED ANALYSIS AND SYSTEM  
ESPECIFICATION.

WILEY - INTERSCIENCE.

R. WENING - SYSTEMS ANALYSIS CHECKLIST

AVERBACH PUBLICHERS.

J. KING AND E. SCHREMS, COST BENEFIT ANALYSIS  
IN INFORMATION SYSTEMS DEVELOPMENT.

## CAPITULO III.

V. BASILI Y M. ZELKOWITZ, ANALYZING MEDIUM SCALE  
SOFTWARE DEVELOPMENT

IEEE

C. WALSTON Y C. FELIX, A. METHOD OF PROGRAMMING  
MEASUREMENTS AND ESTIMATION.

IBM SYSTEMS JOURNAL, VOL. 10.

V. BASILI, MODELS AND METRICS FOR SOFTWARE.

MANAGEMENT AND ENGINEERING

L. PUTMAN, A GENERAL EMPIRICAL SOLUTION TO THE

MACRO SOFTWARE SIZING AND ESTIMATING PROJECT  
IEEE

P. NORDEN, USEFUL TOOLS FOR PROJECT MANAGEMENT  
IEEE

R. ESTERLING, SOFTWARE MAN POWER COSTS: A MODEL  
DATAMATION.

L. PUTMAN SOFTWARE COST ESTIMATING AND LIFE CYCLE  
CONTROL.

IEEE COMPUTER SOCIETY PRESS.

#### CAPITULO V.

E.S. TAYLOR, AN INTERIM REPORT ON ENGINEERING  
DESIGN  
MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

J. DENNIS, MODULARITY  
SPRINGER - VERLAG

E. YOURDAN Y L. CONSTANTINE, STRUCTURED DESIGN  
PRENTICE - HALL.

#### CAPITULO VI.

A. WASSERMAN, INFORMATION SYSTEM DESIGN METHODOLOGY  
IEEE COMPUTER SOCIETY PRESS.

D. ROSS, J. GOODNEOUGH Y C. IRVINE, SOFTWARE  
ENGINEERING : PROCESS, PRINCIPLES AND GOALS  
IEEE COMPUTER SOCIETY PRESS.

B. BOCHM, CHARACTERISTICS OF SOFTWARE QUALITY  
NORTH HOLLAND PUBLISHING Co.

T. McCALL, FACTORS IN SOFTWARE QUALITY  
GENERAL ELECTRIC Co.

I. Mc CABE, A SOFTWARE COMPLEXITY MEASURE  
IEEE COMPUTER SOCIETY PRESS

M. HALSTEAD, ELEMENTS OF SOFTWARE SCIENCE  
NORTH HOLLAND PUBLISHING Co.

#### CAPITULO VII.

P. FREEMAN, THE CONTEXT OF DESIGN  
IEEE COMPUTER SOCIETY PRESS

O. DAHL, E. DIJKSTRA Y C. HOARE, STRUCTURED PROGRAMMING.  
ACADEMIC PRESS.

#### CAPITULO VIII.

J. P. TIEMBLAY Y P.G. SORENSON, AN INTRODUCTION TO  
DATA STRUCTURES WITH APPLICATIONS  
Mc GRAW - HILL .

E. HOROWITZ Y S. SAHNI, FUNDAMENTAL SOFTWARE COMPUTER  
ALGORITHMS.

COMPUTER SCIENCE PRESS

J.D. WARNIER LOGICAL CONSTRUCTION OF PROGRAMS  
VAN NOSTRAND.

A. WASSERMAN, PRINCIPLES OF SYSTEMATIC DATA  
DESIGN AND IMPLEMENTATION.

#### CAPITULO IX.

E. DIJKSTRA, PROGRAMMING CONSIDERED AS A HUMAN ACTIVITY  
NORTH HOLLAND PUBLISHING Co.

N. CHAPIN, A NEW FORMAT FOR FLOW CHARTS  
PRENTICE - HALL.

CAPITULO X

T. PRATT, PROGRAMMING LANGUAGES: DESIGN AND IMPLEMENTATION

PRENTICE - HALL

B. KERNIGHAM Y B. PLAUGER, THE ELEMENTS OF PROGRAMMING STYLE.

Mc GRAW - HILL.

D. VAN TASSEL, PROGRAM STYLE, DESIGN, EFFICIENCY DEBUGGING AND TESTING

PRENTICE - HALL

CAPITULO XI

M. DEUTSCH, VERIFICACION Y VALIDATION

PRENTICE - HALL

W. E. HOWDEN THEORETICAL AND EMPIRICAL STUDIES OF PROGRAM TESTING.

IEEE COMPUTER SOCIETY PRESS.

J. BROWN Y M. LIPOW, TESTING FOR SOFTWARE REALIABILITY.

WILEY

A. BROWN Y W. SAMPSON, PROGRAMMING DEBUGGING AMERICAN ELSEVIER.

E. MILLER, TUTORIAL : AUTOMATED TOOLS FOR SOFTWARE ENGINEERING.

IEEE COMPUTER SOCIETY PRESS.

CAPITULO XII

B. LEINTZ Y E. SWANSON, SOFTWARE MAINTENANIC MANAGMENT.

ADDISON - WILEY

D. FREEDMAN Y B. WEIN BERG. TECHNIQUES OF PROGRAM  
AND SYSTEM MAINTENANCE  
WINTHROP PUBLISHERS.

