1.- Mantenimiento del software

1.1.- Una definición del mantenimiento del software

El mantenimiento del software es mucho más que una "corrección de errores". Podemos describir el mantenimiento describiendo las cuatro actividades que se llevan a cabo tras distribuir el programa.

La primera actividad de mantenimiento es debida a que no es razonable asumir que la prueba del software haya descubierto todos los errores latentes de un gran sistema de software. Durante el uso de cualquier programa, se encontrarán errores, siendo informado el equipo de desarrollo. El proceso que incluye el diagnóstico y la corrección de uno o más errores se denomina ***mantenimiento correctivo***.

La segunda actividad que contribuye a la definición de mantenimiento se produce por el rápido cambio inherente a cualquier aspecto de la informática (hardware, SO, periféricos). Por otro lado la vida útil del software sobrevive al entrono del sistema para el que fue desarrollado. El ***mantenimiento adaptativo*** es una actividad que modifica el software para que interaccione adecuadamente con su entorno cambiante.

La tercera actividad que se puede aplicar a la definición de mantenimiento se produce cuando un software tiene éxito. A medida que se usa el software, se reciben de los usuarios recomendaciones sobre nuevas posibilidades, sobre modificaciones de funciones ya existentes y sobre mejoras en general. Para satisfacer estas peticiones, se lleva a cabo el ***mantenimiento perfectivo o ampliativo***.

La cuarta actividad de mantenimiento se da cuando se cambia el software para mejorar una futura facilidad de mantenimiento o fiabilidad, o para proporcionar una base mejor para futuras mejoras. A menudo denominada ***mantenimiento preventivo***, esta actividad está caracterizada por las técnicas de *ingeniería inversa* y de *reingeniería*.

Las tareas que se dan como parte del mantenimiento adaptativo o perfectivo son las mismas que se aplican durante la fase de desarrollo del proceso de ingeniería del software. Debemos determinar nuevos requisitos, rediseñar, generar código y probar el software.

1.2.- Características del mantenimiento

Las características del mantenimiento se pueden considerar desde tres puntos de vista diferentes:

Mantenimiento estructurado frente al no estructurado

Si solo se dispone del código fuente como elemento de configuración, la actividad de mantenimiento comienza con una tediosa evaluación del código, a menudo complicada por la pobre documentación interna. La estructura del programa, las estructuras de datos, las interfaces, el rendimiento y las limitaciones de diseño son difíciles de descubrir y frecuentemente mal interpretadas. Es difícil asegurar cuales son las ramificaciones de los cambios. Es imposible llevar a cabo pruebas de regresión (repetición de pruebas anteriores) ya que no existe ningún registro de pruebas. Lo que hacemos es un *mantenimiento no estructurado* y pagamos el precio.

Si existe una completa configuración del software, la tarea de mantenimiento comienza con una evaluación de la documentación del diseño. Se determinan las importantes características estructurales, de rendimiento y de la interfaz del software. Se estudia el impacto de las correcciones o modificaciones requeridas y se traza un plan de actuación. Se modifica el diseño y se revisa. Se desarrolla nuevo código fuente, se realizan pruebas de regresión mediante la información contenida en la *especificación de prueba* y se vuelve a lanzar el software.

Esta secuencia de sucesos constituye el *mantenimiento estructurado* y aparece como resultado de una anterior aplicación de una metodología de ingeniería de software.

Costos del mantenimiento

Los costos del mantenimiento en dinero es lo que obviamente más nos interesa. Sin embargo, otros costos, menos tangibles, pueden, en último lugar, ser la causa de muchas preocupaciones.

Costos tangibles:

- ✓ Insatisfacción del cliente cuando una petición de reparación o de modificación aparentemente legítima no se puede atender en un tiempo razonable.
- ✓ Disminución de la calidad global del software debido a los errores latentes que introducen los cambios en el software mantenido.
- ✓ Trastornos en otros esfuerzos de desarrollo al tener que “poner” a trabajar a la plantilla en tareas de mantenimiento.

El esfuerzo empleado en el mantenimiento se puede dividir en actividades productivas (por ej.: análisis y evaluación, modificación del diseño, codificación) y actividades “menos productivas” (por ej.: tratar de comprender lo que es el

código: tratar de interpretar las estructuras de datos, las características de la interfaz, los límites de rendimiento).

Problemas

La mayoría de los problemas asociados con el mantenimiento de software se deben a las deficiencias de la forma en que el software ha sido definido y desarrollado.

Entre los muchos problemas clásicos asociados con el mantenimiento de software se encuentran los siguientes:

- ✓ A menudo es difícil o imposible seguir la evolución del software a través de varias versiones.
- ✓ A menudo es difícil o imposible seguir el proceso por el que se construyó el software.
- ✓ A menudo es excepcionalmente difícil comprender un programa "ajeno".
- ✓ Ese personaje "ajeno", a menudo, no se encuentra cerca para que pueda explicar lo que hizo.
- ✓ No existe una documentación apropiada o está mal preparada.
- ✓ La mayoría del software no ha sido diseñado previendo el cambio.
- ✓ El mantenimiento no se ve como un trabajo atractivo.

1.3.- Facilidad de mantenimiento

La facilidad de mantenimiento se puede definir cualitativamente como la facilidad de comprender, corregir, adaptar y/o mejorar el software.

1.4.- Tareas de mantenimiento

Las tareas asociadas con el mantenimiento del software comienzan mucho antes de que se haga una petición de mantenimiento. Inicialmente, se debe establecer una organización de mantenimiento; se deben prescribir procedimientos de evaluación y de información, y se debe definir una secuencia estandarizada de sucesos para cada petición de mantenimiento. Además, se debe establecer un sistema de registro de información de las actividades de mantenimiento y definir criterios de revisión y de evaluación.

1.5.- Efectos secundarios del mantenimiento

La documentación del diseño y una cuidadosa prueba de regresión ayudan a eliminar errores, pero seguirán apareciendo *efectos secundarios del mantenimiento*.

Categorías de efectos secundarios:

Efectos secundarios sobre el código

Cuando se realizan cambios en el código fuente, es probable que se introduzcan errores. Los siguientes cambios tienen mayor probabilidad de introducir errores que otros:

1. Un subprograma eliminado o cambiado.
2. Cambios para mejorar el rendimiento de ejecución.
3. Modificación de operadores lógicos.
4. Traducción de cambios del diseño en importantes cambios sobre el código.
5. Cambios sobre las pruebas límites.

Efectos secundarios sobre los datos

Los siguientes cambios en los datos, a menudo producen efectos secundarios:

1. Redefinición de constantes.
2. Redefinición de registros o archivos.
3. Aumento o disminución del tamaño de arreglos.
4. Modificación de datos globales.
5. Reorganización de argumentos de E/S o de subprogramas.

Efectos secundarios sobre la documentación

Los efectos secundarios sobre la documentación se dan cuando no se reflejan los cambios de código fuente en la documentación de diseño y en los manuales orientados al usuario.

Siempre que se haga un cambio sobre el flujo de datos, sobre la arquitectura del diseño, sobre los procedimientos (módulos) o sobre cualquier otra característica asociada, se debe actualizar la documentación técnica de soporte.

1.6.- Mantenimiento del código ajeno

Se les denomina código ajeno porque:

1. Los miembros del equipo técnico actual no trabajaron en el desarrollo del programa
2. No se aplicó una metodología de desarrollo y, por tanto, existe un pobre diseño arquitectónico y de datos; la documentación es incompleta y no existe registro de los cambios anteriores.

Documentación del mantenimiento

1.7.- Ingeniería inversa y reingeniería

La *ingeniería inversa* para el software es el proceso de analizar un programa en un esfuerzo de crear una representación del programa de mayor nivel de abstracción que el código fuente. La ingeniería inversa es un proceso de *recuperación de diseño.* Las herramientas de ingeniería inversa extraen la información del diseño de datos, arquitectónico y procedimental.

La *reingeniería,* también denominada *renovación o reclamación*, no solo recupera la información del diseño de un software existente, sino que usa esa información para alterar o reconstruir el sistema existente, en un esfuerzo de mejorar la calidad general. En la mayoría de los casos, el software resultante de la reingeniería reimplementa la función del sistema existente. Pero, al mismo tiempo, el desarrollador añade nuevas funciones y/o mejora el rendimiento general.