INGENIERIA DE SOFTWARE
Tema 7: Mantenimiento del software

Presenta: David Martínez Torres

Universidad Tecnológica de la Mixteca

dtorres@mixteco.utm.mx

Cubo 37

Contenido

1. Aspectos generales
2. Características
3. Categorías de mantenimiento: correctivo, adaptativo, preventivo y perfectivo
4. Reingeniería
5. Conclusiones
6. Referencias

1. Aspectos Generales

- Estrategias de la evolución del software: mantenimiento, reemplazo, evolución arquitectónica y reingeniería del software.
- El mantenimiento de software de un producto consiste en las actividades realizadas sobre la aplicación una vez entregado el producto.
- Se centra en el **cambio.**
- El software es sometido a reparaciones y modificaciones cada vez que se detecta un fallo o se necesita cubrir una nueva necesidad de los usuarios.
- En esta fase recae el mayor porcentaje del coste de un sistema.

1. Aspectos Generales

Un buen sistema no es sólo un conjunto de programas que funcionan.

Debe ser **fácil de mantener**

Documentación esencial(CASE, Computer Assisted Software Engineering)

1. Aspectos Generales

- El glosario de IEEE [IEEE 610] describe el mantenimiento de software como
  - El proceso de modificar un sistema o componente de software entregado para corregir defectos, mejorar el desempeño o algún otro atributo, o adaptarlo al cambio del entorno
- Se estima que el mantenimiento consume entre 40% y 90% de los costos del ciclo de vida de las aplicaciones.
- Lehman afirma como “ley” que si un programa no tiene una adaptación continua a las necesidades existentes, con el tiempo es cada ves menos útil.

Aspectos Generales [Bennet]

- Administración
  - Difícil definir el retorno sobre la inversión
- Proceso
  - Se requiere una amplia coordinación para manejar el flujo de solicitudes de mantenimiento
- Técnica
  - Debe cubrirse todo el impacto de los cambios
  - Las pruebas son muy costosas en comparación con la utilidad de cada cambio
    - Las pruebas concretas son ideales pero costosas
    - Todavía se requieren las pruebas de regresión

Actividades y roles del mantenimiento

- Las actividades del mantenimiento son similares a aquéllas del desarrollo.
- El mantenimiento enfoca simultáneamente cuatro aspectos mayores de la evolución del sistema:
  - Mantener el control sobre las funciones diarias del sistema
  - Mantener el control sobre las modificaciones del sistema
  - Perfeccionar las funciones aceptables existentes
  - Impedir que el desempeño del sistema se degrade a niveles inaceptables.

Tipos de mantenimiento

- Según Lientz,Swason, et. Al [Li], clasifica el mantenimiento en dos:
  - Acciones de mantenimiento que **reparan** defectos
  - Acciones de mantenimiento que **mejoran** la aplicación
- Varios estudios han mostrado que de 60% a 80% de las acciones de mantenimiento son mejoras y no reparaciones.

Tipos de mantenimiento

Tipos de mantenimiento	
Reparación	Correctiva • Identificar defecto y eliminarlo Adaptable • Cambios obtenidos al operar los cambios en el sistema, hardware o DBMS
Mejoras	Perfeccionamiento • Cambios que resultan de las solicitudes de los usuario Preventivas • Cambios hechos al software para facilitar el mantenimiento

Mantenimiento correctivo

- A pesar de las pruebas y verificaciones que aparecen en etapas anteriores del ciclo de vida del software, los programas pueden tener defectos.
- El mantenimiento correctivo tiene por objetivo localizar y eliminar los posibles defectos de los programas.
- Un defecto en un sistema es una característica del sistema con el potencial de causar un fallo.
- Un fallo ocurre cuando el comportamiento de un sistema es diferente del establecido en la especificación.

Mantenimiento correctivo

- Entre otros, los fallos en el software pueden ser de:
- **Procesamiento**: por ejemplo, salidas incorrectas de un programa.
- **Rendimiento**: por ejemplo, tiempo de respuesta demasiado alto en una búsqueda de información.
- **Programación**: por ejemplo, inconsistencias en el diseño de un programa.
- **Documentación**: por ejemplo, inconsistencias entre la funcionalidad de un programa y el manual de usuario.

Mantenimiento adaptativo

- Este tipo de mantenimiento consiste en la modificación de un programa debido a cambios en el entorno (hardware o software) en el cual se ejecuta.
- Estos cambios pueden afectar al sistema operativo (cambio a uno más moderno), a la arquitectura física del sistema informático (paso de una arquitectura de red de área local a Internet/Intranet) o al entorno de desarrollo del software (incorporación de nuevos elementos o herramientas como ODBC).

Mantenimiento adaptativo

- La envergadura del cambio necesario puede ser muy diferente: desde un pequeño retoque en la estructura de un módulo hasta tener que reescribir prácticamente todo el programa para su ejecución en un ambiente distribuido en una red.

Mantenimiento adaptativo

- Los cambios en el entorno software pueden ser de dos clases:
  - En el entorno de los datos, por ejemplo, al dejar de trabajar con un sistema de ficheros clásico y sustituirlo por un sistema de gestión de bases de datos relacionales.
  - En el entorno de los procesos, por ejemplo, migrando a una nueva plataforma de desarrollo con componentes distribuidos, Java, ActiveX, etc.

Mantenimiento adaptativo

- El mantenimiento adaptativo es cada vez más usual debido principalmente al cambio, cada vez más rápido, en los diversos aspectos de la informática: nuevas generaciones de hardware cada dos años, nuevos sistemas operativos -ó versiones de los antiguos- que se anuncian regularmente, y mejoras en los periféricos o en otros elementos del sistema. Frente a esto, la vida útil de un sistema software puede superar fácilmente los diez años [Pressman, 1993].

Mantenimiento perfectivo

- Cambios en la especificación, normalmente debidos a cambios en los requisitos de un producto software, implican un nuevo tipo de mantenimiento llamado perfectivo.
- La casuística es muy variada. Desde algo tan simple como cambiar el formato de impresión de un informe, hasta la incorporación de un nuevo módulo aplicativo.
- Podemos definir el mantenimiento perfectivo como el conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario.

Mantenimiento perfectivo

- Algunos autores dividen este tipo de mantenimiento en dos:
  - Mantenimiento de Ampliación: orientado a la incorporación de nuevas funcionalidades.
  - Mantenimiento de Eficiencia: que busca la mejora de la eficiencia de ejecución.
- Este tipo de mantenimiento aumenta cuando un producto software tiene éxito comercial y es utilizado por muchos usuarios, ya que cuanto más se utiliza un software, más peticiones de los usuarios se reciben demandando nuevas funcionalidades o mejoras en las existentes

Mantenimiento preventivo

- Este último tipo de mantenimiento consiste en la modificación del software para mejorar sus propiedades (por ejemplo, aumentando su calidad y/o su mantenibilidad) sin alterar sus especificaciones funcionales.
- Por ejemplo, se pueden incluir sentencias que comprueben la validez de los datos de entrada, reestructurar los programas para mejorar su legibilidad, o incluir nuevos comentarios que faciliten la posterior comprensión del programa.

Mantenimiento preventivo

- Este tipo de mantenimiento es el que más partido saca de las técnicas de ingeniería inversa y reingeniería.
- En algunos casos se ha planteado el
  - Mantenimiento para la Reutilización, consistente en modificar el software (buscando y modificando componentes para incluirlos en bibliotecas) para que sea mas fácilmente reutilizable. En realidad este tipo de mantenimiento es preventivo, especializado en mejorar la propiedad de reusabilidad del software.

Reingenieria del software

- Se refiere a reimplementar los sistemas heredados para hacerlos más mantenibles. La reingeniería comprende la redocumentación del sistema, la organización y reestructura del sistema, la traducción del sistema a un lenguaje de programación más moderno y la modificación y actualización de la estructura y los valores de los datos del sistema.

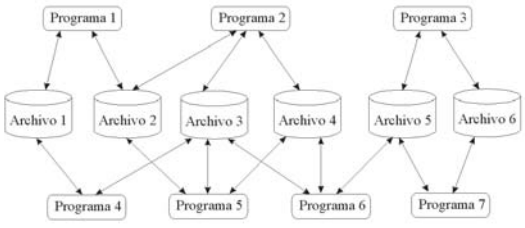
Sistemas heredados

- Son sistemas usados por la organización para sus negocios, lo cuales deben mantenerse
- Normalmente sus costos de mantenimiento se incrementan
- Hay millones de líneas de código fuente, generalmente escritas en COBOL o FORTRAN.

... Sistemas heredados

- Desarrollados antes de que el uso de las técnicas de ingeniería de software estuvieran difundidas. No están estructurados ni documentados.
- Incrustados de conocimiento crítico del negocio, el cual puede no estar documentado en ningún lugar. No hay especificación del sistema.
- El riesgo de re-implementar estos sistemas es muy alto.

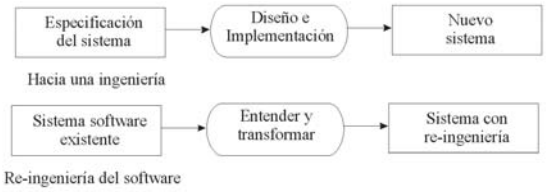
Solicitud de sistemas.



Cuando hacer re-ingeniería

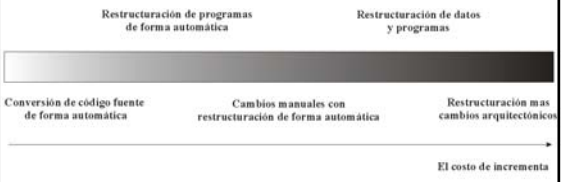
- Cuando los cambios del sistema son necesarios en solo una parte.
- Cuando el soporte de hardware o software se hacen obsoletos.
- Cuando las herramientas para soportar la reestructuración están disponibles.

Re-ingeniería y desarrollo



El proceso de reingeniería

Enfoques de la Re-ingeniería



Re-ingeniería.- Factores de costo

- La calidad del software que será sujeto de re-ingeniería
- La herramienta disponible para re-ingeniería
- La amplitud de la conversión de datos requerida
- La disponibilidad de personal experto para hacer re-ingeniería.

Ventajas de la re-ingeniería

- Reduce riesgo. El re-desarrollo de software esencial para una empresa, es una actividad de alto riesgo debido al papel crítico del software en el negocio
- Reduce costos. Algunas medidas sugieren que el costo de re-ingeniería es significativamente menor que el costo de re-desarrollo, quizá hasta 4 veces menos.

Traducción de código fuente

- Implica convertir el código de un lenguaje (o de una versión de lenguaje) a otro. Por ejemplo de FORTRAN A C
- Puede ser necesario debido a:
  - Actualización de la plataforma de hardware.
  - Falta de conocimiento del personal de mantenimiento
  - Cambio en las políticas de la empresa.
  - Falta de sw., de apoyo
- Solo se puede hacer si hay un traductor automático disponible

Re-estructuración de programas

- El mantenimiento tiende a corromper la estructura de un programa. Cada vez es más difícil de entenderlo.
- El programa puede re-estructurarse automáticamente quitándole ramificaciones incondicionales.
- Las condiciones pueden simplificarse para hacerlas más leíbles.

Simplificación de una condición

-- condición compleja
If not (A > B **and** (C < D **or not** (E > F))) ...

-- condición simplificada
If (A <= B **and** (C>= D **or** E > F) ...

Problemas de la reestructuración automática de programas

- Los problemas con la re-estructuración son:
  - Pérdida de comentarios
  - Pérdida de documentación
  - fuertes demandas de cómputo
- La re-estructuración se dificulta donde hay una pobre modularización o donde los componentes relacionados están dispersos por todos lados.
- El entendimiento de programas que manejan datos puede no mejorarse con la reestructuración

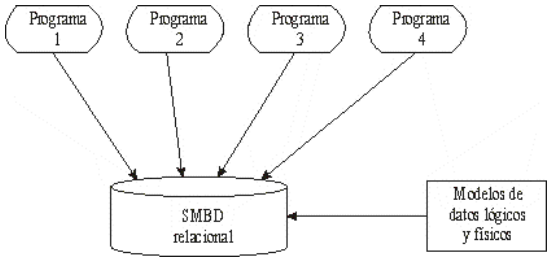
Ingeniería inversa

- Analizar software con el fin de recuperar su diseño y especificación
- El programa mismo no cambia
- La entrada a este proceso es el código fuente o ejecutable.
- Puede ser parte de un proceso de re-ingeniería pero también puede usarse para re-especificar un sistema para su re-implantación
- Algunas herramientas, (browsers, generadores de referencias cruzadas) pueden usarse en este proceso.

Re-ingeniería de datos

- Proceso de analizar y reorganizar las estructuras y algunas veces, los valores de los datos de un sistema para hacerlo más comprensible.
- Puede ser parte del proceso de migración de un sistema de archivos a un DBMS o el cambio de una base a otra.
- El objetivo es crear un medio administrable de datos

Medio ambiente de la administración de datos



Problemas con los datos

- Los usuarios finales quieren datos en sus máquinas en lugar de sistemas de archivos. Necesitan descargar estos datos de una DBMS.
- Los sistemas pueden tener que procesar muchos más datos de los considerados originalmente por los diseñadores.
- Datos redundantes pueden almacenarse en diferentes formatos y en diferentes lugares en el sistema.

Problemas con los datos.

- Problemas con los nombres de los datos
- Problemas con la longitud de los campos.
- Problemas en la organización de los registros.
- Literales fuertemente codificados
- No hay diccionario de datos.

Inconsistencias en los valores de los datos

- Inconsistencia en los valores de default
- Inconsistencia en las unidades
- Inconsistencia en las reglas de validación
- Inconsistencia en la representación semántica.
- Inconsistencia en el manejo de valores negativos

9. Referencias

1. Somerville, Ian (2002) ***"****Ingeniería de software,"* 6a edición. Addison Wesley.
2. Pressman, S Roger (1998) "*Ingeniería del Software: Un enfoque práctic*o," 4a edición McGraw-Hill.
3. Braude Eric J. (2003) "Ingeniería de Software Una perspectiva orientada a objetos," Alfaomega
4. Ince (1994) Detalles de ISO
5. Oskarrson y Glass (1995) Detalles de ISO

40

- ¿Preguntas?
- Gracias!

41