

Fundamentos de Ingeniería del Software



Capítulo 10. Mantenimiento del software

Mantenimiento del software.

Estructura



1. Introducción
2. Tipos de mantenimiento
3. Costes del mantenimiento
4. Dificultades del mantenimiento
5. El proceso de mantenimiento en el ciclo de vida del sw.
6. Métodos de mantenimiento del software
 - Reingeniería
 - Redocumentación
 - Ingeniería inversa
 - Ingeniería inversa de procesos (comprensión de programas)
 - Identificación y recopilación de los componentes funcionales
 - Asignar valor semántico a los componentes funcionales
 - Reconstrucción de programas
 - Ingeniería inversa de ficheros y BD
 - Ingeniería inversa y reingeniería de interfaces de usuario
7. Mantenibilidad o facilidad de mantenimiento del sw.

Mantenimiento del software.

Bibliografía



- (Piattini et *al.* 98) M. Piattini, J. Villalba, F. Ruiz, I. Fernández, M. Polo, T. Bastanchury, M.A. Martínez. "Mantenimiento del software" Ed. Ra-Ma. 1998.
- (Piattini et *al.* 04) M. Piattini, José A. Calvo-Manzano, J. Cervera, L. Fernández. "Análisis y diseño detallado de Aplicaciones Informáticas de Gestión". Ed. Ra-Ma. 1996. Capítulo 15.
- (Piattini et *al.* 96) M. Piattini, José A. Calvo-Manzano, J. Cervera, L. Fernández. "Análisis y diseño detallado de Aplicaciones Informáticas de Gestión". Ed. Ra-Ma. 1996. Capítulo 16.

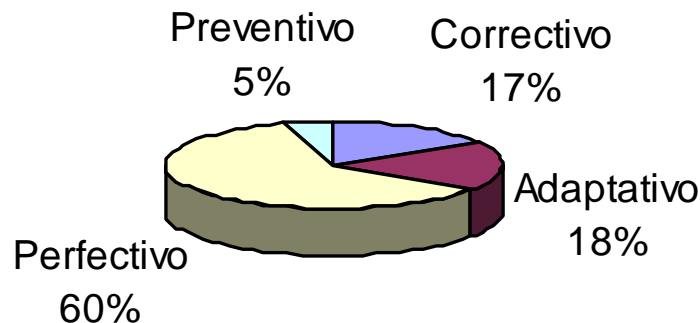
1. Introducción

- “El mantenimiento del sw. es la modificación de un producto sw. después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de entorno” (IEEE 1219)
- Es la parte más costosa del ciclo de vida del sw.:
60-90% del coste total (y coste creciente)
⇒ *El coste relativo de reparar un defecto aumenta en las últimas etapas del ciclo de vida (de 1 a 100)*
- En algunas empresas coste del 95% ⇒ *Barrera del mantenimiento*
(no se pueden desarrollar nuevos productos sw.)

2. Tipos de mantenimiento

- Correctivo
- Adaptativo
- Perfectivo
 - *Mantenimiento de ampliación*
 - *Mantenimiento de eficiencia*
- Preventivo
 - *Mantenimiento para la reutilización*

Coste de mantenimiento



Tipos de mantenimiento (II)



- El mantenimiento perfectivo aumenta cuando un producto software tiene éxito comercial y es usado por muchos usuarios: se reciben más peticiones solicitando mejoras o nuevas funcionalidades.
- El mantenimiento preventivo consiste en la modificación del software para mejorar sus propiedades sin alterar sus especificaciones funcionales (p.ej. aumentando su calidad o su mantenibilidad):
 - incluir sentencias que validen los datos de entrada
 - reestructurar los programas para mejorar su legibilidad
 - incluir nuevos comentarios
 - ...
- El preventivo es el tipo de mantenimiento que más usa las técnicas de reingeniería e ingeniería inversa.
- Mantenimiento para la reutilización: mantenimiento preventivo que trata de mejorar la propiedad de reutilización (reusabilidad) del software.

3. Costes del mantenimiento



- Oportunidades de desarrollo que se pierden.
- Insatisfacción del cliente cuando no se puede atender en un tiempo aceptable una petición de reparación que parece razonable.
- Los errores ocultos que se introducen al cambiar el sw. durante el mantenimiento reducen la calidad global del producto.
- Perjuicio en otros proyectos de desarrollo cuando la plantilla tiene que dejarlos, total o parcialmente, para atender peticiones de mantenimiento.

Coste de las actividades de mantenimiento

<i>Categoría</i>	<i>Actividad</i>	<i>% Tiempo</i>
Comprensión del sw. y de los cambios a realizar	Estudiar las peticiones	18%
	Estudiar la documentación	6%
	Estudiar el código	23%
Modificación del sw.	Modificar el código	19%
	Actualizar la documentación	6%
Realización de pruebas	Diseñar y realizar pruebas	28%

(Piattini et al. 98)

⇒ nótese cómo la comprensión del software y de los cambios supone casi un 50% del coste total de mantenimiento

4. *Dificultades del mantenimiento*



- Una de las principales, las aplicaciones heredadas (*legacy code*), que siguen funcionando, pero en muchas ocasiones adolecen de:
 - diseño pobre de las estructuras de datos
 - restricciones de tamaño y espacio de almacenamiento
 - herramientas desfasadas, sin métodos
 - documentación escasa
 - una o varias migraciones a nuevas plataformas
 - múltiples modificaciones para adaptarlos o mejorarlos
 - mala codificación
 - lógica defectuosa
 - desarrolladores no localizables
 - ¿*desechar el sw. y reescribirlo?* No siempre factible:
 - gran carga financiera de su desarrollo
 - necesidad de amortización
- ⇒ sw. que sigue funcionando con baja calidad

Dificultades del mantenimiento (II)



- Ausencia de métodos
(se realiza de forma *ad hoc*).
- Ausencia de documentación.
- No captura adecuada de requisitos \Rightarrow mayores esfuerzos de mantenimiento futuros.
- Cambio tras cambio, los programas tienden a ser menos estructurados.
- No existen registros de pruebas \Rightarrow imposibilidad de pruebas de regresión.
- Problemas de gestión
 \Rightarrow considerado "trabajo poco creativo",
es asignado a las personas con menos experiencia

5. El proceso de mantenimiento en el ciclo de vida del sw.



- Proceso ppal. de mantenimiento en el std. IEEE 12207.
- Actividades:
 - Implementación del proceso.
 - Análisis de problemas y modificaciones.
 - Implementación de las modificaciones.
 - Revisión y aceptación del mantenimiento.
 - Migración.
 - Retirada del sw.

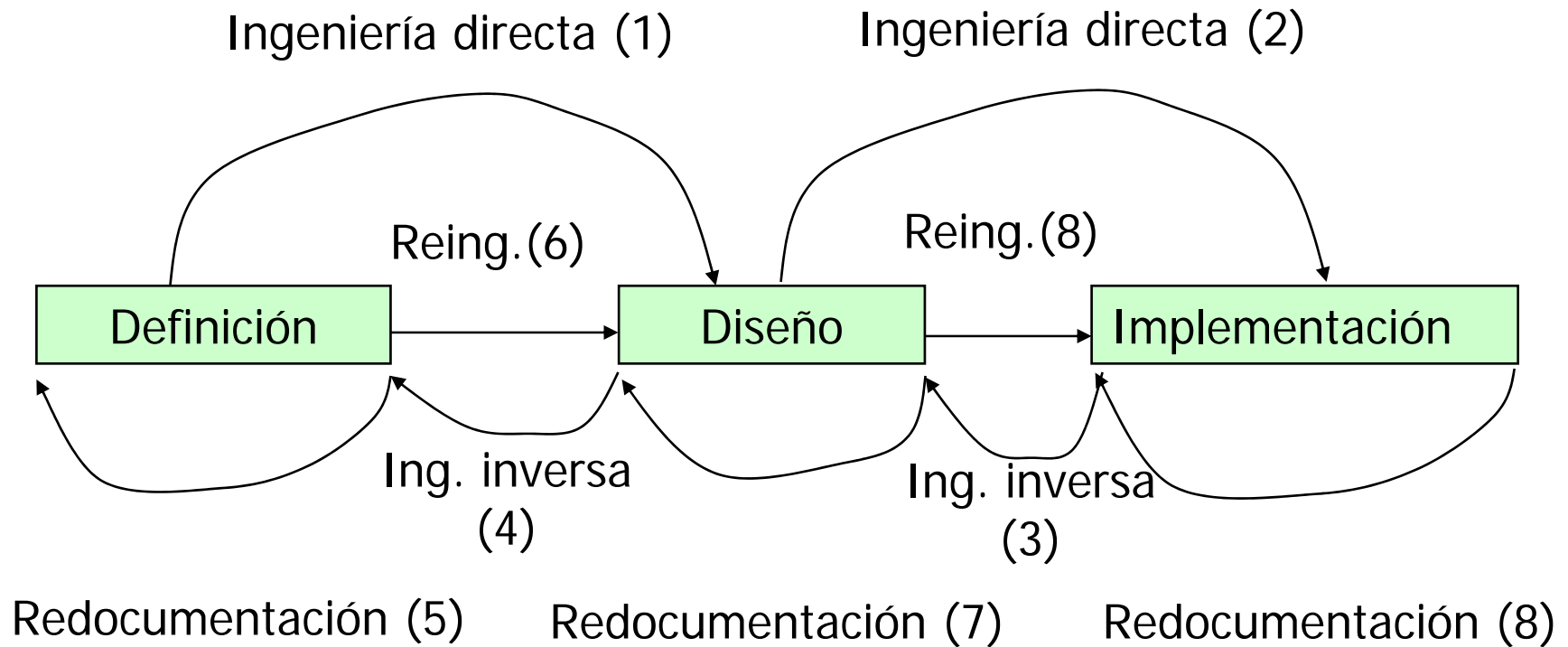
6. *Métodos de mantenimiento del software*



A menudo se utilizan conjuntamente:

- *Ingeniería inversa*: análisis de un sistema para identificar sus componentes y las relaciones entre ellos, así como para crear representaciones del sistema en otra forma o en un nivel de abstracción más elevado.
- *Reingeniería*: examen y modificación del sistema para reconstruirlo en una nueva forma.
- *Reestructuración del software*: consiste en la modificación del software para hacerlo más fácil de entender y cambiar o menos susceptible de incluir errores en cambios posteriores.
- *Transformación de programas*: técnica formal de transformación de programas

Ingeniería directa, inversa, reingeniería y redocumentación




Reingeniería



- Objetivo: métodos para reconstruir el sw.:
 - reprogramarlo
 - redocumentarlo
 - rediseñarlo
 - rehacer alguna/s característica/s del producto
- Reingeniería: “la modificación de un producto sw., o de ciertos componentes, usando para el análisis del sistema existente técnicas de ingeniería inversa y, para la etapa de reconstrucción, herramientas de ingeniería directa”

Redocumentación

(Pressman 02) p.546 (Pressman 06) p.910



- a) Si el sistema funciona y la redocumentación consume muchos recursos, tal vez mejor no redocumentar.
- b) Si es preciso actualizar la documentación, pero recursos limitados, puede ser útil “documentar cuando se modifica”. Con el tiempo, se formará una colección de información interesante.
- c) Si el sistema es fundamental para la organización, redocumentar por completo. Se puede reducir la documentación al mínimo.

Ingeniería inversa



- Ingeniería inversa: “el proceso de construir especificaciones abstractas del código fuente de un sistema heredado, de manera que estas especificaciones puedan ser utilizadas para construir una nueva implementación del sistema *hacia delante*”
 - Ingeniería inversa. Beneficios (Piattini et al. 96):
 - Reducir la complejidad del sistema
 - Generar vistas alternativas
 - Recuperar la información perdida (cambios que no se documentaron en su momento)
 - Detectar efectos laterales
 - Facilitar la reutilización
- ⇒ Ingeniería inversa: *El pto. de partida no es necesariamente el código fuente* (Piattini et al. 96)

Ingeniería inversa de procesos (comprensión de programas)

```
char *f (char *c) {
    unsigned long i, lon;
    char *x = (char *) malloc (100);
    lon = strlen(c);
    if (c[0]=='\'' ) {
        for (i=0; i<lon-1; i++) x[i]=c[i+1]; x[i]='\0';
    } else
    if (c[0]=='"') {
        for (i=0; i<lon-1; i++) x[i]=c[i+1]; x[i]='\0';
    } else for (i=0; i<lon; i++)
        x[i]=c[i];
    if ((x[strlen(x)-1]=='\'' ) ||
        (x[strlen(x)-1]=='\"'))
        x[strlen(x)-1]='\0';
    return x;
}
```

Ingeniería inversa de procesos (comprensión de programas) (II)

```
#define COMILLA_SIMPLE  '\''
#define COMILLA_DOBLE  '\"'

/*  Autor: Juan Gómez Montijo.
    Entradas: una cadena.
    Devuelve: la misma cadena sin comillas ni al principio ni al final, si es que las
              tenía. */

char * QuitaComillas(char *cadena){
    unsigned long i, l;
    char *resultado=(char *) malloc(100);
    /* Quitamos la comilla final, si la hay */
    if ((cadena[strlen(cadena)-1]==COMILLA_SIMPLE ||
        cadena[strlen(cadena)-1]==COMILLA_DOBLE))
        cadena[strlen(cadena)-1]='\0';
    /* Pasamos la cadena a una auxiliar, quitando la comilla inicial si la hay */
    if ((cadena[0]==COMILLA_SIMPLE || cadena[0]==COMILLA_DOBLE)){
        for (i=0; i<strlen(cadena)-1; i++) {
            resultado[i]=cadena[i+1];
        }
    } else
        for (i=0; i<strlen(cadena); i++) {
            resultado[i]=cadena[i];
        }
    resultado[i]='\0';
    return resultado;
}
```

Ingeniería inversa de procesos

Tareas necesarias (Piattini et al. 98)



1. Identificación y recopilación de los componentes funcionales del sistema
 - Rutinas, variables, constantes, tipos de datos, TAD, objetos, llamadas a funciones, etc.
 - Centrarse sólo en los componentes “sustanciales”
2. Asignar significado a los componentes anteriores
 - ⇒ no se realizan secuencialmente

Identificación y recopilación de los componentes funcionales



- Muy subjetiva e intuitiva.
- Algunas ideas:
 - cada componente suele ocupar un módulo
 - los componentes suelen aparecer próximos unos a otros
 - las series de componentes funcionales suelen aparecer junto a muchos comentarios
 - los identificadores de los componentes funcionales suelen constar de muchos caracteres
- Algunos problemas:
 - sinonimia
 - polisemia
 - comentarios no actualizados

Asignar valor semántico a los componentes funcionales



- Se recorren las sentencias del componente, para confirmar su calidad de componente funcional.
 - Análisis estático (p.ej., creación de diagramas de flujo)
 - Análisis dinámico (ejecución del programa)
- Detectar y registrar bucles infinitos, código inalcanzable, etc.
 - que no se corrigen en esta fase, sólo se documentan
 - se corrigen después, en la fase de reingeniería

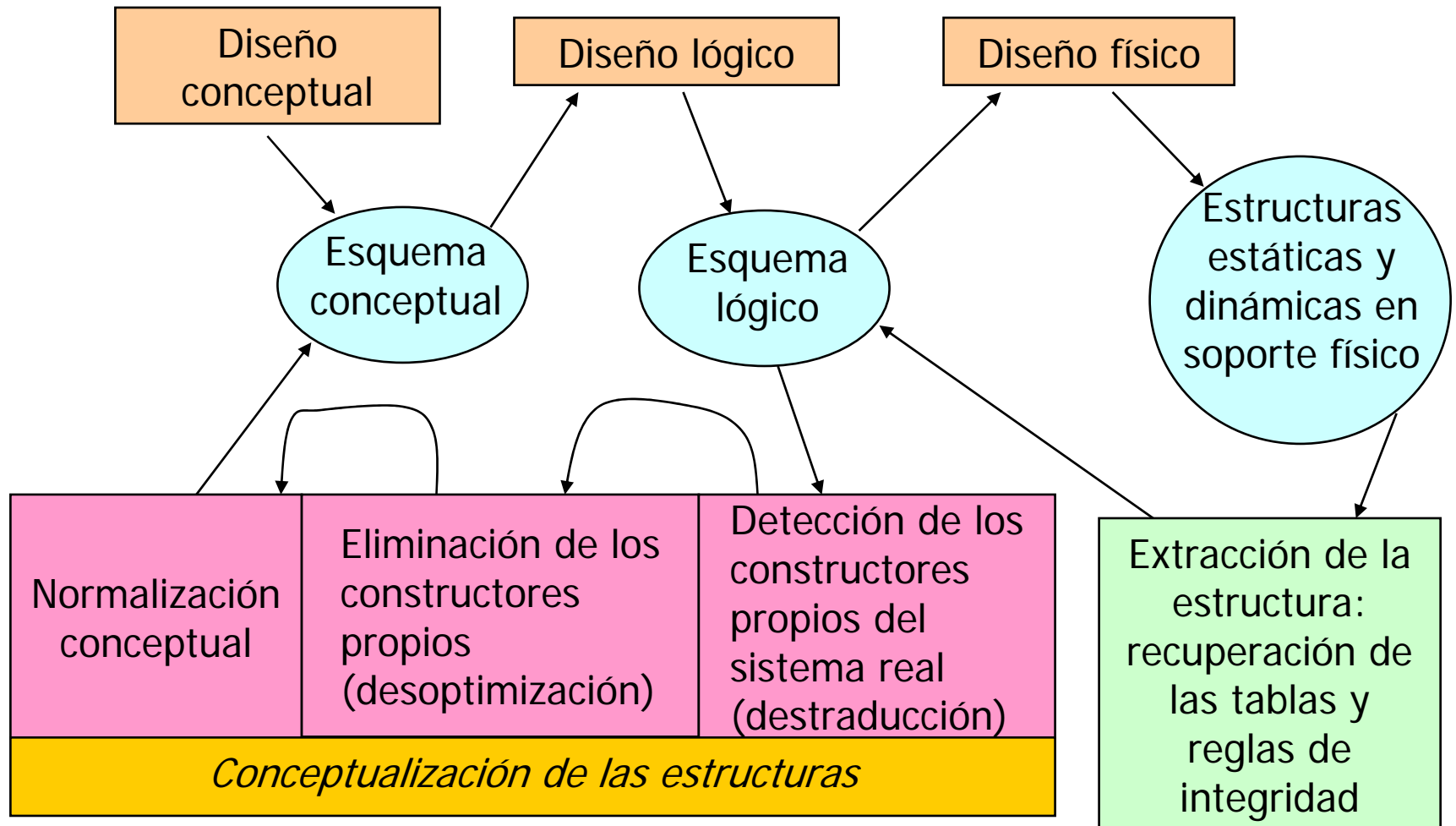
Reconstrucción de programas



- A partir de los productos de ingeniería inversa se construye el programa mediante técnicas de ingeniería directa.
- Reestructuración de datos
 - eliminar sinonimias y polisemias
- Reestructuración de procesos
 - transformar el código no estructurado en código estructurado

⇒ en un diagrama de flujo estructurado, es posible hacer transformaciones sucesivas hasta que su complejidad ciclomática se iguale a 1 (Piattini et al. 96) p.547

Ingeniería inversa de ficheros y BD



Ingeniería inversa de ficheros.

Extracción de la estructura



- Considerar cada fichero como una posible tabla, y cada campo del fichero como un campo de la tabla.
 - Determinar un conjunto de campos que puedan ser clave primaria de sus respectivos ficheros (buscar ID, #).
 - Determinar las claves ajenas.
 - Determinar los ficheros que no pueden tratarse como tablas (aquellos sin claves).
 - Buscar generalizaciones:
 - grandes grupos de claves ajenas.
 - valores repetidos de atributos en una tabla.
 - datos con valores mutuamente excluyentes.
- (Piattini et al. 98)

Ingeniería inversa y reingeniería de interfaces de usuario



Adaptar aplicaciones a las necesidades de los usuarios, respetando su lógica anterior:

1. Recopilación de documentación, manuales de usuario, etc.
2. Entrevistas a distintos grupos de usuarios, y observación de sus métodos de trabajo.
3. Uso del sistema por el propio equipo de mantenimiento
⇒ se puede modificar el código para, p.ej., introducir contadores
4. Reconstrucción y redocumentación de la interfaz.

7. Mantenibilidad o facilidad de mantenimiento del sw.



- Medida cualitativa de la facilidad de comprender, corregir, adaptar y/o mejorar el SW.
- “Facilidad con que un sistema o componente sw. puede ser modificado para corregir defectos, mejorar el rendimiento u otros atributos, o adaptarse a un cambio de entorno”
- Muy ligada a la calidad del sw.
⇒ métricas de mantenibilidad =
métricas de calidad
- También ligada a la complejidad del sw.

Métricas para mantenibilidad (Mc Call) (Piattini et al. 98)

