Mantenimiento de software

1. Introducción
2. Conceptos fundamentales
3. Evolución del software y sus leyes
4. El proceso de mantenimiento
5. Técnicas para el mantenimiento del software
6. Métricas de mantenimiento

INTRODUCCIÓN

El **Mantenimiento** constituye con creces la mayor parte del esfuerzo total del desarrollo, siendo una actividad que consume muchos recursos

Actividades que suponen costes directos

- Comprender el software y entender los cambios que se deben realizar
- Modificar el software y actualizar la documentación
- Volver a realizar las pruebas del software (pruebas de regresión)

Actividades que suponen costes indirectos

- **Costes de oportunidad**

  Desarrollos que se han de posponer o que nunca se realizarán porque los recursos disponibles están dedicados a las tareas de mantenimiento

- **Insatisfacción del cliente**

  Cuando una reparación o modificación que parece razonable no se puede atender en un tiempo aceptable

- **Errores ocultos**

  Introducidos al cambiar el software durante el mantenimiento y que reducen la calidad del producto

- **Prejuicio causado a otros proyectos de desarrollo**

  Cuando el personal tiene que dejarlos, total o parcialmente, para atender peticiones de mantenimiento

CONCEPTOS FUNDAMENTALES

¿Qué es el mantenimiento del software?

Facilidad de mantenimiento

Mantenimiento y calidad

Propiedades de la facilidad de mantenimiento

CONCEPTOS FUNDAMENTALES

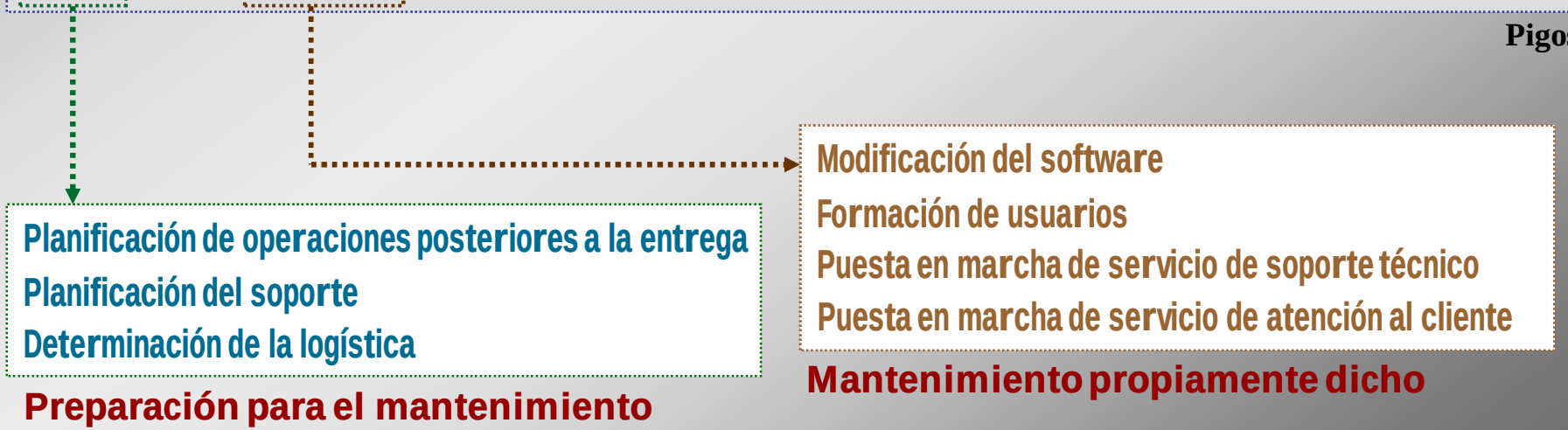
¿Qué es el mantenimiento del software?

Mantenimiento del software es la modificación de un producto software después de la entrega para corregir fallos, para mejorar su rendimiento u otros atributos, o para adaptar el producto a un entorno modificado

Estándar 1219 de IEEE

Mantenimiento del software son la totalidad de las actividades necesarias para proporcionar un soporte rentable al software. Estas actividades se desarrollan tanto antes como después de la entrega

Pigoski



CONCEPTOS FUNDAMENTALES

Facilidad de mantenimiento

Facilidad de mantenimiento es la disposición de un sistema o componente software para ser modificado con objeto de corregir fallos, mejorar su funcionamiento u otros atributos, o adaptarse a cambios en el entorno

Glosario IEEE de términos de ingeniería del software

Factores que afectan directamente a la facilidad de mantenimiento

- **El proceso de desarrollo**
  - La facilidad de mantenimiento debe formar parte integral del proceso de desarrollo y las técnicas usadas deben ser lo menos intrusivas posible con el software existente
  - Problemas :
  - Mejorar la facilidad de mantenimiento
  - Convencer a los responsables de la ganancia que se obtendrá
- **La documentación**
  - En ocasiones ni la documentación ni las especificaciones de diseño están disponibles
  - Problemas :
  - Los costes se incrementan debido al tiempo requerido para entender el diseño antes de modificarlo
  - La responsabilidad del mantenimiento se transfiere a una organización diferente

CONCEPTOS FUNDAMENTALES

- **La comprensión de los programas**

  La causa básica de los altos costes de mantenimiento es la presencia de obstáculos a la comprensión humana de los programas y sistemas existentes

  Problemas :

  La información disponible es incomprensible, incorrecta o insuficiente

  La complejidad del software o de la propia naturaleza de la aplicación

  Malas interpretaciones sobre el por qué de algunas decisiones de diseño

Factores que favorecen la facilidad de mantenimiento

- Las técnicas de programación estructurada
- La aplicación de metodologías de ingeniería del software
- Seguimiento de estándares

CONCEPTOS FUNDAMENTALES

Mantenimiento y calidad

La facilidad de mantenimiento es un atributo de calidad, que se subdivide en cinco características, cada una de las cuales se puede medir con métricas específicas

Estándar ISO/IEC 9126

- **Fácil de analizar**
  Facilidad para diagnosticar deficiencias o causas de fallos, o para identificar partes que se deben modificar
- **Fácil de cambiar**
  El software debe permitir cambios especificados previamente en el diseño, el código y la documentación
- **Estable**
  Los efectos inesperados de las modificaciones deben tener impacto mínimo
- **Fácil de probar**
  Debe permitir evaluar las modificaciones
- **Conforme**
  Debe satisfacer los estándares o normas sobre facilidad de mantenimiento

CONCEPTOS FUNDAMENTALES

Propiedades de la facilidad de mantenimiento

La facilidad de mantenimiento se puede valorar por la combinación de dos propiedades

Un sistema software es reparable si permite la corrección de sus defectos con una cantidad de trabajo limitada y razonable

- Le afecta especialmente la cantidad y tamaño de los módulos que componen el software
- El incremento excesivo del número de módulos no implica un producto más reparable
- Buscar un punto de equilibrio con la estructura de módulos para garantizar que el software sea reparable

Un sistema software es flexible si permite la introducción de cambios para satisfacer nuevos requisitos, es decir, si puede evolucionar

- El software es mucho más fácil de cambiar o incrementar en sus funciones que un producto físico
- La flexibilidad se ve disminuida con cada nueva versión, ya que cada versión complica su estructura
- La aplicación de técnicas y metodologías apropiadas puede minimizar el impacto de la flexibilidad
- La flexibilidad es una característica tanto del producto como del proceso de construcción

EVOLUCIÓN DEL SOFTWARE Y SUS LEYES

La **evolución del software** es el conjunto de actividades de programación que se orientan a generar una nueva versión de un software a partir de una versión anterior operativa

Lehman y Ramil

Leyes de la evolución del software

En el ámbito de las ciencias y la ingeniería

Una ley debe entenderse como una característica común a muchos fenómenos o que se presenta con regularidad

La elaboración completa de las 8 leyes de Lehman se prolongó durante más de 20 años (1968-1996)

- **Cambio continuo**
  Un software que se utiliza debe adaptarse continuamente, en caso contrario, se hace progresivamente menos satisfactorio
- **Complejidad creciente**
  A medida que evoluciona un software, su complejidad se incrementa, a menos que se trabaje para mantenerla o reducirla

EVOLUCIÓN DEL SOFTWARE Y SUS LEYES

- **Autorregulación**
  El proceso de evolución del software se autorregula con una distribución de medidas de atributos de producto y procesos cercana a la normal
- **Conservación de la estabilidad organizativa (velocidad de trabajo invariable)**
  La velocidad de estabilidad global efectiva media en un sistema en evolución no varía a lo largo del ciclo de vida del producto
- **Conservación de familiaridad**
  Conforme un sistema evoluciona, todo lo asociado con él debe mantener el dominio de su contenido y comportamiento para lograr evolución satisfactoria
- **Crecimiento continuo**
  El contenido funcional de los sistemas debe aumentar continuamente para mantener la satisfacción del usuario durante su tiempo de vida
- **Declive de la calidad**
  La calidad de los sistemas declinará, a menos que se mantengan y adapten rigurosamente a los cambios del entorno operativo
- **Retroalimentación del sistema**
  Los procesos evolutivos constituyen sistemas de retroalimentación multinivel, mutibucle y multiagente, y deben tratarse como tales para lograr mejora significativa sobre cualquier base razonable

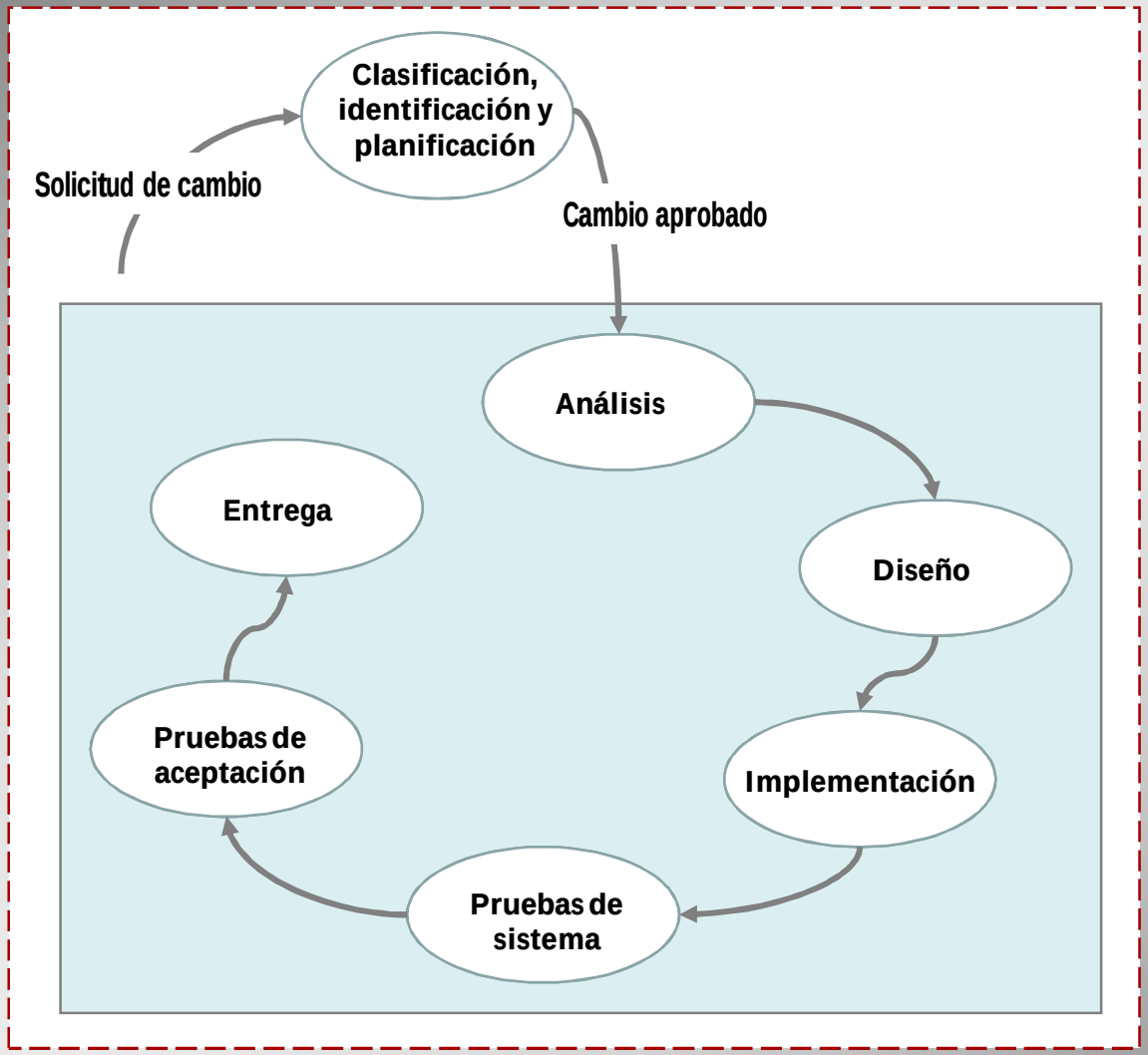
EL PROCESO DE MANTENIMIENTO

Actividades de mantenimiento

Clasificación del mantenimiento

EL PROCESO DE MANTENIMIENTO

Actividades de mantenimiento



Un cambio en el software sigue un microciclo completo de ingeniería, al que de be preceder una actividad específica de clasificación, identificación y toma de decisiones respecto a los cambios que deben hacerse en el software

EL PROCESO DE MANTENIMIENTO

Causas que desencadenan las actividades de mantenimiento

- **Eliminación de defectos del producto software**
  - El cliente encuentra un fallo en el software que le han entregado
  - Ejemplo
  - Un cálculo incorrecto en una aplicación de contabilidad
  - La organización que lo desarrolló es responsable de encontrar el defecto y corregirlo

- **Adaptación del producto software a nuevos requisitos**
  - El cliente tiene nuevas necesidades
  - Ejemplo
  - Nueva legislación fiscal que haga necesario adaptar el software
  - A este tipo de cambios se les suele denominar perfectivos

- **Inclusión de mejoras en el diseño**
  - Mejorar el diseño del software como prevención de cambios futuros
  - Ejemplo
  - Los desarrolladores consideran que la estructura interna del software es difícil de comprender

EL PROCESO DE MANTENIMIENTO

Clasificación del mantenimiento

- **Mantenimiento correctivo**
  Modificaciones a un producto software después de la entrega para corregir defectos descubiertos
- **Mantenimiento adaptativo**
  Modificaciones a un producto software después de la entrega para permitir que ese producto se pueda seguir utilizando en un entorno diferentes
- **Mantenimiento perfectivo**
  Modificación de un producto software después de la entrega para mejorar el rendimiento o la facilidad de mantenimiento

Lientz y Swanson

+

- **Mantenimiento preventivo**
  Orientado a prevenir errores que puedan surgir en el futuro

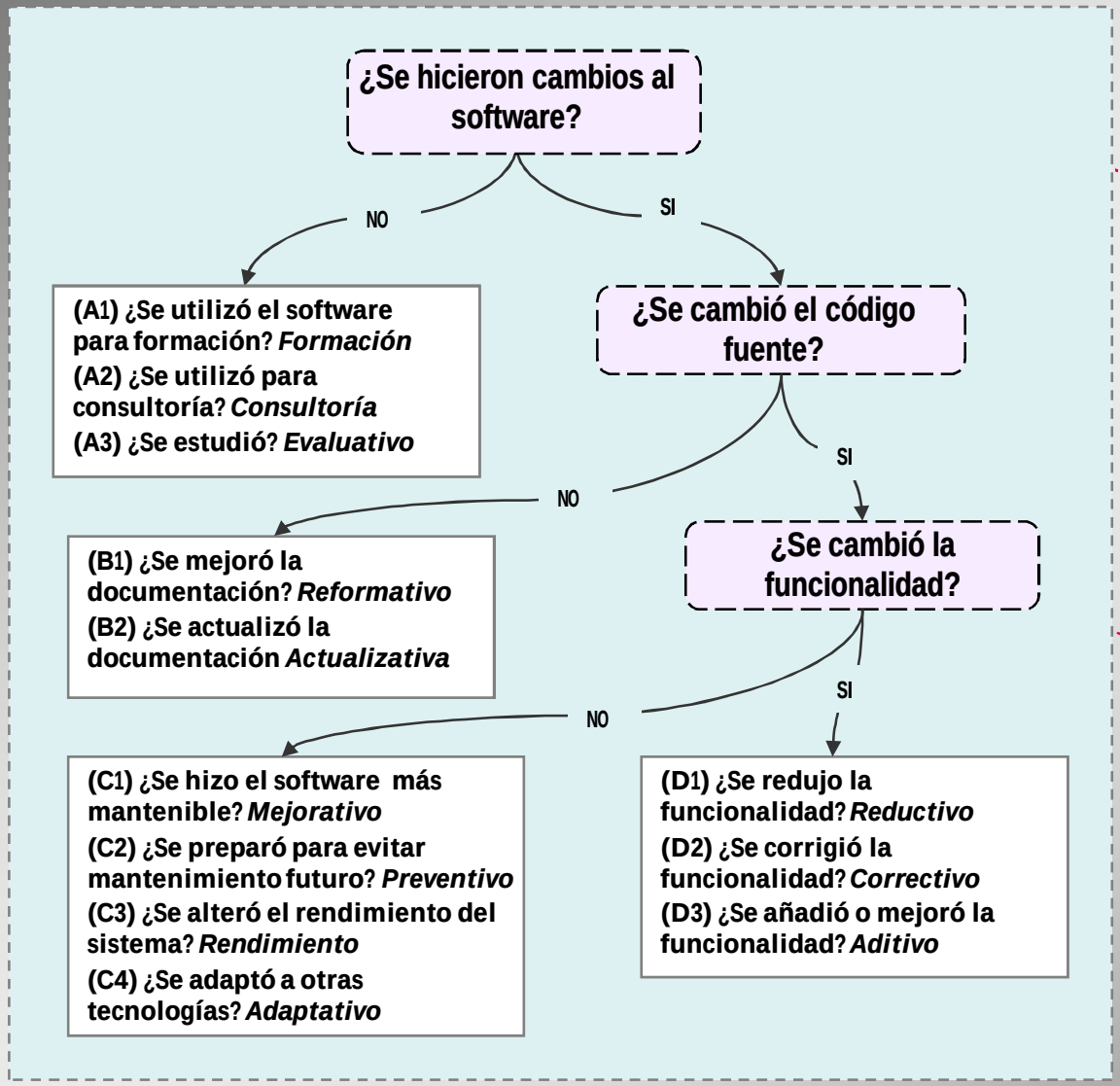
Estándar ISO/IEC 14764

+

- **Mantenimiento de emergencia**
  Engloba los cambios que deben realizarse sin planificación previa, pues resultan esenciales para mantener un sistema en operación

Estándar 1219 de IEEE

EL PROCESO DE MANTENIMIENTO



Esta clasificación permite comprender el amplio abanico de actividades que se pueden categorizar como mantenimiento

Todas estas actividades deben ser objeto de planificación y control en la ingeniería del software

Ned Chapin

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Ingeniería inversa

Reingeniería

Reestructuración

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Soluciones técnicas

La **ingeniería inversa** consiste en analizar un sistema para identificar sus componente y las relaciones entre ellos, así como en elaborar nuevas representaciones del sistema, generalmente de más alto nivel

La **reingeniería** es la modificación de componentes software mediante el uso de técnicas de ingeniería inversa para su análisis y herramientas de ingeniería directa para su reconstrucción

La **reestructuración (refactoring)** es un cambio de representación de un producto software, pero dentro del mismo nivel de abstracción

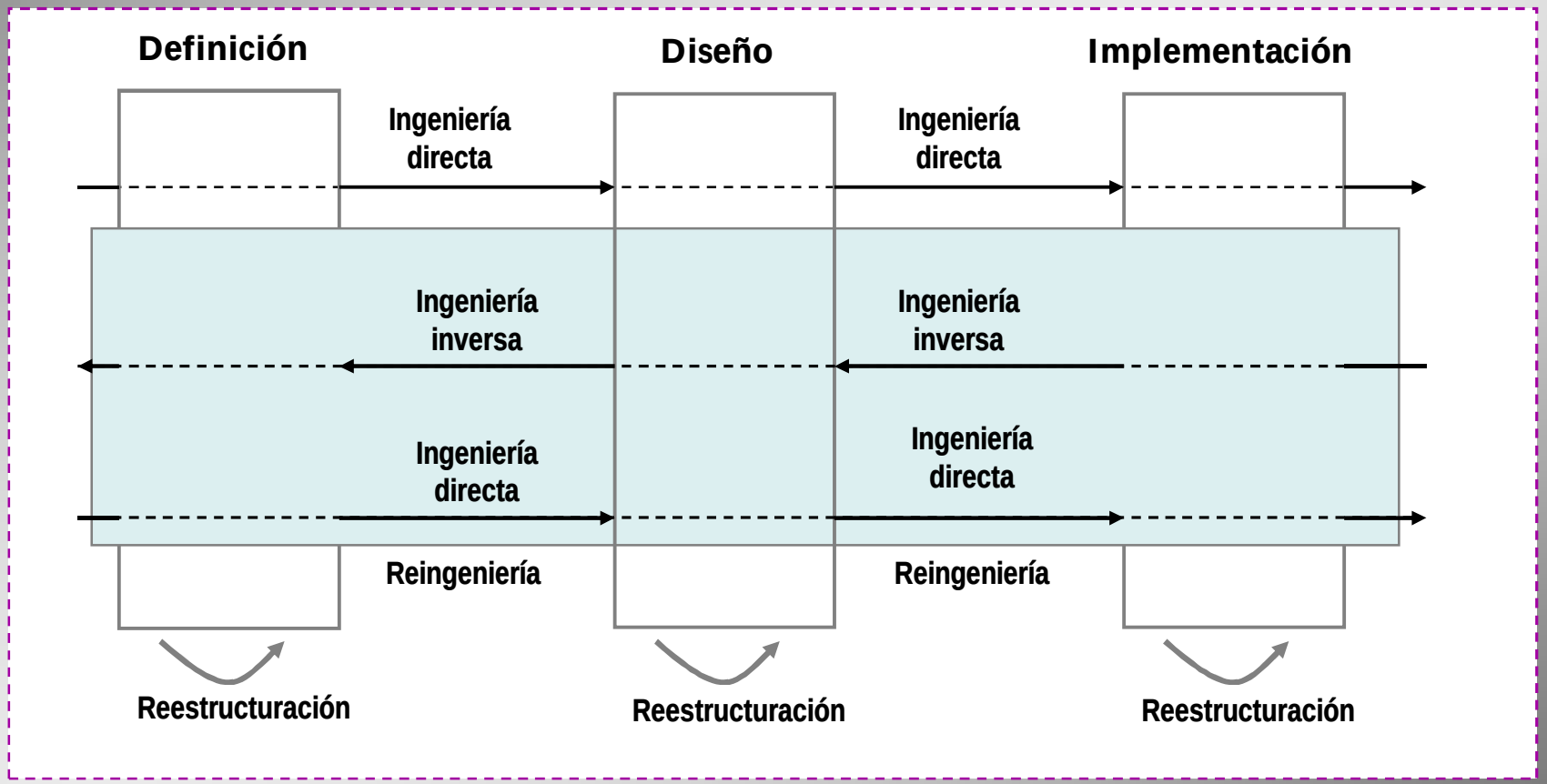
Objetivo de estas técnicas

Proporcionar métodos para reconstruir el software, bien reprogramándolo, volviéndolo a documentar, rediseñándolo o rehaciendo características del producto

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Diferencia entre ellas

Cuál es el origen y cuál el destino de las mismas



TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Ingeniería inversa

- **Características**

  Las especificaciones se pueden volver a usar para construir una nueva implementación del sistema
  La aplicación de la ingeniería inversa nunca cambia la funcionalidad del software
  Permite obtener productos que indican como se ha construido el software

- **Beneficios**

  Reduce la complejidad del sistema
  Genera diferentes alternativas para el diseño, distintas de las del diseño original
  Recupera y/o actualiza información perdida
  Detecta efectos colaterales
  Facilita la reutilización

- **Tipos**

  Ingeniería inversa de datos
  Ingeniería inversa de lógica o de proceso
  Ingeniería inversa de interfaces

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Ingeniería inversa de datos

Se aplica sobre algún código de bases de datos para obtener los modelos relacionales, o también sobre el modelo relacional para obtener el diagrama conceptual

Se usa para

- Modificar una base de datos
- Migrar a un nuevo sistema de gestión de bases de datos
- Crear el modelo de datos del sistema software

Ingeniería inversa de procesos

Se aplica sobre el código de un programa para extraer su lógica, o sobre un documento de diseño para obtener documentos de análisis o de requisitos

Se usa para

- Entender mejor la aplicación y regenerar el código
- Migrar la aplicación a un nuevo sistema operativo
- Generar o completar la documentación
- Comprobar que el código cumple las especificaciones de diseño

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

¿Cómo se lleva a cabo la ingeniería inversa de procesos?

1. Buscar el programa principal
2. Ignorar inicializaciones de variables y sentencias similares
3. Inspeccionar la primera rutina de llamada y determinar si es importante
4. Inspeccionar las rutinas invocadas por la primera rutina del programa principal y examinar las que parezcan importantes
5. Repetir los pasos 3 y 4 en el resto del software
6. Recopilar esas rutinas importantes (*componentes funcionales*)
7. Asignar significado a cada componente funcional según el siguiente proceso
   (a). Explicar qué hace cada componente funcional en el conjunto del sistema
   (b). Explicar qué hace el sistema a partir de los diferentes componentes funcionales

Ingeniería inversa de interfaces de usuario

Se aplica para obtener los modelos y especificaciones que sirvieron de base para la construcción de un programa, y tomarlas como punto de partida en procesos de ingeniería directa que permitan modificar su interfaz

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Reingeniería

- **Características**

  Es un proceso mediante el cual se mejora un software existente
  Hace uso de técnicas de ingeniería inversa y reestructuración de código
  Las tareas que hay que llevar a cabo requieren tiempo y esfuerzo

- **Beneficios**

  Facilita el mantenimiento del software
  Permite la adaptación a nuevos estándares, tecnologías, etc.
  Es la alternativa más económica frente a un nuevo desarrollo del sistema

- **Es una buena opción cuando**

  La aplicación tiene fallos frecuentes difíciles de localizar
  La aplicación es poco eficiente, pero realiza la funcionalidad esperada
  Existen dificultades para integrar la aplicación con otros sistemas
  El software de la aplicación es de poca calidad
  No se dispone de personal suficiente para realizar todas las modificaciones que pueden surgir
  No es fácil hacer las pruebas oportunas a todos los cambios que se realizarán
  El mantenimiento de la aplicación consume muchos recursos
  Es necesario incluir nuevos requisitos, si bien los fundamentales se mantienen

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Procesos implicados en la reingeniería

- **Análisis de inventario**

  Estudio de la antigüedad, importancia de la aplicación en el negocio y facilidad de mantenimiento actual, etc., para estudiar la posible conveniencia de la reingeniería

- **Reestructuración de documentos**

  Proceso en el que se puede elegir una de las siguientes opciones

  - Obviar la documentación de los módulos estáticos que no va a sufrir cambios
  - Documentar sólo los que se va a modificar
  - Documentar toda la información del sistema, si ésta es fundamental para el negocio

- **Ingeniería inversa**

  Se extraen modelos de alto nivel de abstracción que ayuden a la comprensión de la aplicación, para poder modificarla

  - Los modelos extraídos sirven de punto de partida para el siguiente proceso
  - Se deben almacenar en un repositorio
  - Se conforma la documentación de análisis y diseño para facilitar el posterior mantenimiento

- **Reestructuración de código y de datos y/o aplicación de ingeniería directa**

  A la luz de los resultados de la ingeniería inversa, se reestructuran el código y los datos, o se aplian ténicas de ingeniería directa para rehacer la aplicación

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Reestructuración

Características

Modifica una parte del sistema sin alterar los resultados de otras etapas
No requiere conocimiento total del sistema, únicamente de la parte a modificar

Ejemplo

Reestructuración de un programa para mejorar su legibilidad

Refactoring

Refactoring es el proceso de cambiar el software de un sistema para mejorar su estructuración interna pero sin alterar su comportamiento externo

Características

Limpia el código para que sea más fácil de entender y modificar
Mejora el diseño del software
Ayuda a encontrar errores ocultos que no han salido a la luz todavía

TÉCNICAS PARA EL MANTENIMIENTO DEL SOFTWARE

Técnicas de refactoring

- **(Re-)composición de métodos**
  Se basan en acortar métodos demasiado largos, sustituir llamadas a los métodos por su código, etc.
- **Movimiento de características entre clases**
  Reasignan responsabilidades de una clase a otra
  Separar una clase en varias, eliminar una clase, introducir nuevos métodos en una clase, etc.
- **Reorganización de datos**
  Permiten facilitar el trabajo con datos
  Crear métodos *get* y *set* para consultar atributos los propios atributos de una clase, etc.
- **Simplificación de expresiones condicionales**
  Facilitan la comprensión, depuración y mantenimiento del software mediante la simplificación de estructuras condicionales
  Dividir una condición compleja, eliminar expresiones condicionales redundantes, etc.
- **Simplificación de las llamadas a los métodos**
  Simplifican la interfaz de una clase para que sea más fácil de usar
  Cambiar nombre de métodos, eliminar o agregar parámetros a los métodos, etc.
- **Reorganización de la jerarquía de generalización**
  Mueven métodos a lo largo de la jerarquía de herencia
  Añadir método de subclase a una superclase, añadir constructores a las superclases, etc.

MÉTRICAS DE MANTENIMIENTO

Métricas del producto

Métricas relacionadas con el proceso

MÉTRICAS DE MANTENIMIENTO

En el mantenimiento, la medición se considera como un atributo de calidad el software

Para determinar los costes de mantenimiento

Se mide la facilidad de mantenimiento
Se utiliza esa medida como factor de evaluación

Si la medida se realiza durante el desarrollo

Ayuda a determinar en qué grado se incorpora la facilidad de mantenimiento al proceso de producción

Si la medida se realiza cuando se ha completado el desarrollo

Sirve para evaluar el impacto de un cambio o para realizar un análisis comparativo entre varias propuestas

Características

No evalúan el coste de realizar un cambio en el sistema software
Evalúan aspectos de complejidad y calidad

Mayor calidad mayor facilidad de mantenimiento

Mayor complejidad menor facilidad de mantenimiento

MÉTRICAS DE MANTENIMIENTO

Métricas del producto

Índice de madurez del software

Proporciona una indicación de la estabilidad del software

$$IMS = \frac{M_T - (F_a + F_m + F_e)}{M_T}$$

M_T = número de módulos en la versión actual
F_m = número de módulos en la versión actual que han sido modificados
F_a = número de módulos en la versión actual que han sido añadidos
F_e = número de módulos de la versión anterior eliminados de la versión actual

Características

El esfuerzo de mantenimiento será menor

Si el IMS se aproxima a 1, el producto empieza a estabilizarse

Requiere el registro histórico detallado de los cambios en el software

Se basa en contar los cambios entre versiones

MÉTRICAS DE MANTENIMIENTO

Métricas de calidad de la documentación

No existen muchas métricas conocidas

Una de las formas de documentación más habituales de un programa son los comentarios que se incluyen en el código

Densidad de comentarios (para Java)

Densidad de comentarios= Nº JavaDocs / LoC

LoC = número total de líneas de código

Características

Los comentarios explican aspectos del código que no se explican pos si mismos
Cuanto mayor sea la densidad de comentarios mayor facilidad de mantenimiento

Otras métricas

Las que proceden del campo de la ligüística y miden la legibilidad de los documentos

MÉTRICAS DE MANTENIMIENTO

Métricas de calidad del diseño

Métricas para evaluar la calidad de los sistemas orientados a objetos

Aunque las interrelaciones entre los subsistemas son necesarias, los diseño fuertemente interrelacionados son muy rígidos, poco reutilizables y difíciles de mantener

Robert Martin, 1995

Medida de inestabilidad

$$I = \frac{C_e}{C_a + C_e}$$

C_e = número de clases dentro del subsistema que dependen de clases de otros subsistemas

C_a = número de clases de otos subsistemas que dependen de clases del subsistema en estudio

Esta medida varía entre 1 (máxima inestabilidad) y 0 (máxima estabilidad)
Los sistemas extremadamente estables no admiten cambios

→ Máxima facilidad de mantenimiento

Medida de abstracción

$$A = \frac{\text{Clases abstractas}}{\text{Clases totales}}$$

Los sistemas serán más estables cuantas más clases abstractas incluyan
Varía entre 0 (completamente concreto) y 1 (completamente abstracto)

Diseños ni demasiado abstractos ni demasiado inestables

MÉTRICAS DE MANTENIMIENTO

Índice de facilidad de mantenimiento

Mide la facilidad de mantenimiento del producto software teniendo en cuenta que el mantenimiento puede dividirse en tres tareas

Tareas

- Comprensión de los cambios que deben hacerse
- Realización de las modificaciones necesarias
- Pruebas de los cambios realizados

Características

El índice se modela mediante un polinomio en el que se ponderan las tareas

Cuanto mayor sea el índice mayor facilidad de mantenimiento del sistema

Los coeficientes que aparecen en el polinomio no son únicos

Los parámetros se ajustan empíricamente

MÉTRICAS DE MANTENIMIENTO

Métricas relacionadas con el proceso

Miden la facilidad de mantenimiento evaluando el proceso de fabricación del producto

- Tiempo de desarrollo del producto
- Esfuerzo de desarrollo del producto
- Número y tipo de recursos empleados
- Número de peticiones de acciones correctivas
- Coste del proceso

- **Número de defectos encontrados en la fase de pruebas**

  Este número está correlacionado positivamente con el número de defectos del software que se encontrarán durante su explotación

- **Métricas relacionadas con las reparaciones**

  Miden la frecuencia de fallos debidos a efectos laterales producidos después de una modificación

$$X = \frac{1 - A}{B}$$

A = número de fallos debidos a efectos laterales detectados y corregidos

B = número total de fallos corregidos

- **Tiempo medio de reparación**

  Tiempo medio necesario para reparar un fallo y hacer que el sistema vuelva a funcionar normalmente