Ingeniería de software II

Mantenimiento

¿Porqué surge el mantenimiento?

- Éste comienza casi de inmediato. El software se libera a los usuarios finales y, en cuestión de días, los reportes de errores se filtran de vuelta hacia la organización de ingeniería de software.

- En semanas, una clase de usuarios indica que el software debe cambiarse de modo que pueda ajustarse a las necesidades especiales de su entorno.

- Y en meses, otro grupo corporativo, que no quería saber nada del software cuando se liberó, ahora reconoce que puede ofrecerle beneficios inesperados. Pero…necesitará algunas mejoras para hacer que funcione en su mundo.

Fuente: Pressman Cap 29

Una hipótesis

"Mucho del software del que dependemos en la actualidad tiene en promedio una antigüedad de 10 a 15 años. Aun cuando dichos programas se crearon usando las mejores técnicas de diseño y codificación conocidas en la época [y muchas no lo fueron], se produjeron cuando el tamaño del programa y el espacio de almacenamiento eran las preocupaciones principales.

Luego migraron a nuevas plataformas, se ajustaron para cambios en máquina y tecnología de sistema operativo, y aumentaron para satisfacer las necesidades de los nuevos usuarios, todo sin suficiente preocupación por la arquitectura global. El resultado es estructuras pobremente diseñadas, pobre codificación, pobre lógica y pobre documentación de los sistemas de software que ahora debemos seguir usando..."

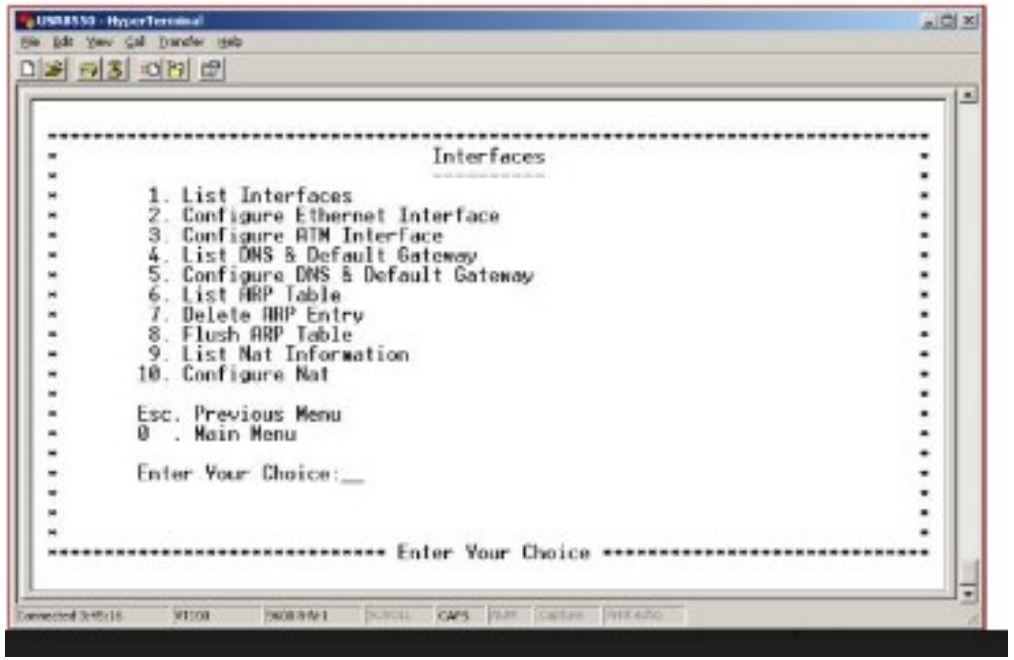
Osborne y Chikofsky [Osb90]

Fuente: Pressman Cap 29

Características del Mantenimiento

En general las características de los sistemas son:

- ✓ Viejos.
- ✓ Sin metodología ni documentación.
- ✓ Sin modularidad.



Fuente: Pfleeger Cap. 11

Mantenimiento

- Atención del sistema a lo largo de su evolución después que el sistema se ha entregado.

  El mantenimiento de software es el proceso de modificar, actualizar y cambiar el software para satisfacer las necesidades del cliente. Es una actividad amplia que incluye: corregir errores, mejorar las capacidades, eliminar funciones obsoletas y optimizar otras.

- A esta fase se la llama “Evolución del Sistema”.

- En ocasiones debe realizarse mantenimiento a sistemas “heredados”. ¿Qué problemas podemos encontrar en los sistemas heredados?

Fuente: Pfleeger Cap. 11

Mantenimiento

- Es necesario evaluar cuándo es conveniente cerrar el ciclo de vida de ese sistema y reemplazarlo por otro.

  La decisión se toma en función del costo del ciclo de vida del viejo proyecto y la estimación del nuevo proyecto

  En ocasiones la complejidad del sistema crece por los cambios.

Fuente: Pfleeger Cap. 11

Dinámica de la evolución de los programas

- Manny Lehman y sus colaboradores realizaron análisis detallados de software de grado industrial y de sistemas en general con la intención de desarrollar una “teoría unificada para evolución del software”.

Fuente: Sommerville Cap,21

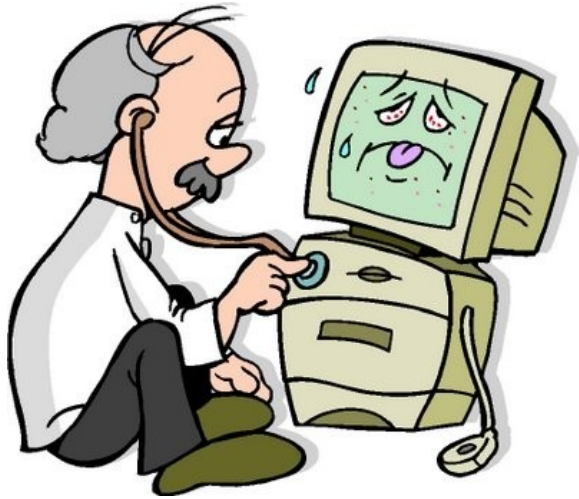
Leyes de Lehman

Cambio continuado	Un programa que se usa en un entorno real necesariamente debe cambiar o se volverá progresivamente menos útil en ese entorno.
Complejidad creciente	A medida que un programa en evolución cambia, su estructura tiende a ser cada vez más compleja. Se deben dedicar recursos extras para preservar y simplificar la estructura.
Evolución prolongada del programa	La evolución de los programas es un proceso autorregulativo. Los atributos de los sistemas, tales como tamaño, tiempo entre entregas y el número de errores documentados, son aproximadamente invariantes para cada entrega del sistema.
Estabilidad organizacional	Durante el tiempo de vida de un programa, su velocidad de desarrollo es aproximadamente constante e independiente de los recursos dedicados al desarrollo del sistema.
Conservación de la familiaridad	Durante el tiempo de vida de un sistema, el cambio incremental en cada entrega es aproximadamente constante.
Crecimiento continuado	La funcionalidad ofrecida por los sistemas tiene que crecer continuamente para mantener la satisfacción de los usuarios.
Decremento de la calidad	La calidad de los sistemas comenzará a disminuir a menos que dichos sistemas se adapten a los cambios en su entorno de funcionamiento.
Realimentación del sistema	Los procesos de evolución incorporan sistemas de realimentación multiagente y multibucle y éstos deben ser tratados como sistemas de realimentación para lograr una mejora significativa del producto.

Fuente: Sommerville Cap,21

Mantenimiento

- Solucionar errores
- Añadir mejoras
- Optimizar



- Esto provoca altos costos adicionales

EL FENÓMENO DE LA

"**BARRERA DE MANTENIMIENTO**"

Fuente: Pfleeger Cap. 11

Mantenimiento - Características

- Su consecuencia es la disminución de otros desarrollos.
- Pueden existir efectos secundarios sobre código, datos, documentación.
- Las modificaciones pueden provocar disminución de la calidad total del producto.
- Las tareas de mantenimiento generalmente provocan reiniciar las fases de análisis, diseño e implementación.
- Involucra entre un 40% a 70% del costo total de desarrollo.
- Los errores provocan insatisfacción del cliente.

Fuente: Pfleeger Cap. 11

Mantenimiento - ¿Por qué es problemático?

- No es un trabajo atractivo
- No siempre en el diseño se prevén los cambios
- Es difícil comprender código ajeno, más aún sin documentación o con documentación inadecuada

Fuente: Pfleeger Cap. 11

Actividades de Mantenimiento

Debe utilizarse un mecanismo para realizar los cambios que permita: identificarlos, controlarlos, implementarlos e informarlos

El proceso de cambio se facilita si en el desarrollo están presentes los atributos calidad como , modularidad, documentación interna del código fuente y de apoyo

Fuente: Pfleeger Cap. 11

Mantenimiento – Ciclo de mantenimiento

- Análisis:

  comprender el alcance y el efecto de la modificación

- Diseño:

  rediseñar para incorporar los cambios

- Implementación:

  recodificar y actualizar la documentación interna del código

- Prueba:

  revalidar el software

- Actualizar la documentación de apoyo
- Distribuir e instalar las nuevas versiones

Fuente: Pfleeger Cap. 11

Facilidades en el desarrollo para ayudar al mantenimiento

- Análisis:

  Señalar principios generales, armar planes temporales, especificar controles de calidad, identificar posibles mejoras, estimar recursos para mantenimiento

- Diseño arquitectónico:

  Claro, modular, modificable, con notaciones estandarizadas

- Diseño detallado:

  Notaciones para algoritmos y estructuras de datos, especificación de interfaces, manejo de excepciones, efectos colaterales

- Implementación:

  Indentación, comentarios de prólogo e internos, codificación simple y clara

- Verificación:

  Lotes de prueba y resultados

Fuente: Pfleeger Cap. 11

¿Quién realiza el mantenimiento?

- El equipo que desarrolla un sistema no siempre es el que se utiliza para mantener el sistema una vez que esté operativo.
- A menudo, un equipo de mantenimiento independiente se emplea para garantizar que el sistema funcione correctamente.
- El equipo de mantenimiento involucra a los usuarios, operadores o representantes del cliente se acercan al equipo de mantenimiento con un comentario o problema. Los analistas o programadores determinan qué partes del código se ve afectado, el impacto en el diseño y los recursos probables (incluido el tiempo y esfuerzo) para realizar los cambios necesarios.

Fuente: Plfeeger

Tareas del equipo de mantenimiento

- ✓ Comprender el sistema
- ✓ Localizar información en la documentación del sistema
- ✓ Mantener actualizada la documentación del sistema
- ✓ Ampliar las funciones existentes para adaptarse a requisitos nuevos o cambiantes
- ✓ Agregar nuevas funciones al sistema
- ✓ Encontrar la fuente de fallas o problemas del sistema
- ✓ Localizar y corregir fallos
- ✓ Responder preguntas sobre la forma en que funciona el sistema
- ✓ Reestructuración del diseño y los componentes del código
- ✓ Rescribir componentes de diseño y código
- ✓ Eliminar componentes de diseño y código que ya no son útiles
- ✓ Gestionar los cambios en el sistema a medida que se realizan

Fuente:

Tipos de Mantenimiento

Mantenimiento correctivo

- Diagnóstico y corrección de errores.

Mantenimiento adaptativo

- *Modificación del software para interaccionar correctamente con el entorno.*

Mantenimiento perfectivo

- *Mejoras al sistema.*

Mantenimiento preventivo

- *Se efectúa antes que haya una petición, para facilitar el futuro mantenimiento. Se aprovecha el conocimiento sobre el producto.*

Fuente: Pfleeger Cap. 11

Mantenimiento correctivo

- El mantenimiento correctivo de software es una acción reactiva que se realiza cuando se detecta un fallo o error en una pieza de software. El objetivo es restablecer el funcionamiento óptimo del sistema y minimizar el impacto del problema en la producción.
- Suele realizarse lo más rápido posible. Los encargados del mantenimiento deben detectar la causa del fallo, decidir si se puede reparar o no y aplicar la solución elegida.

Fuente:

Mantenimiento adaptativo

- El Mantenimiento adaptativo de software tiene que ver con las tecnologías cambiantes, así como con las políticas y reglas relacionadas con su software.
- Las cuales incluyen cambios en el sistema operativo, almacenamiento en la nube, hardware, etc. Cuando se realizan estos cambios, su software debe adaptarse para cumplir adecuadamente los nuevos requisitos y continuar funcionando bien

Fuente:

Mantenimiento perfectivo

- El mantenimiento perfectivo de software es un tipo de mantenimiento que se realiza para mejorar la eficiencia o rendimiento de un sistema. Este mantenimiento se centra en características que mejoran la experiencia del usuario a través de mejoras funcionales, en respuesta a los comentarios de los clientes.
- Los usuarios pueden ver la necesidad de nuevas características o requisitos que les gustaría ver en el software para convertirlo en la mejor herramienta disponible para sus necesidades. El mantenimiento perfectivo de software tiene como objetivo ajustar el software agregando nuevas características según sea necesario y eliminando características que son irrelevantes o no efectivas en el software dado. Este proceso mantiene el software relevante a medida que el mercado y las necesidades del usuario cambian.

Fuente:

Mantenimiento preventivo

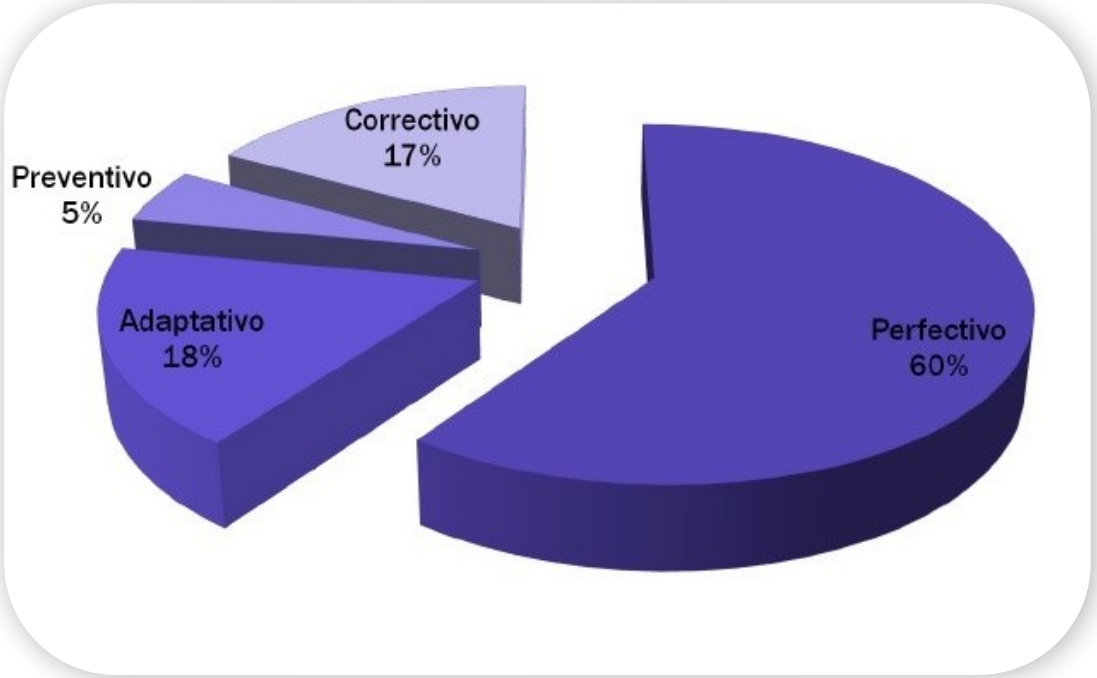
El mantenimiento preventivo de software es un conjunto de actividades planificadas y realizadas regularmente para prevenir posibles fallos en el futuro. Estas actividades incluyen:

- Realizar cambios necesarios
- Actualizaciones
- Prevenir fallas en el sistema operativo, antivirus o programas ofimáticos
- Instalar y configurar un cortafuego para impedir accesos no autorizados de terceros
- Programas antimalware para impedir que el equipo se infecte con malware
- Activar y configurar los puntos de restauración del sistema para poder volver a un estado anterior en caso de algún desastre

Fuente:

Mantenimiento

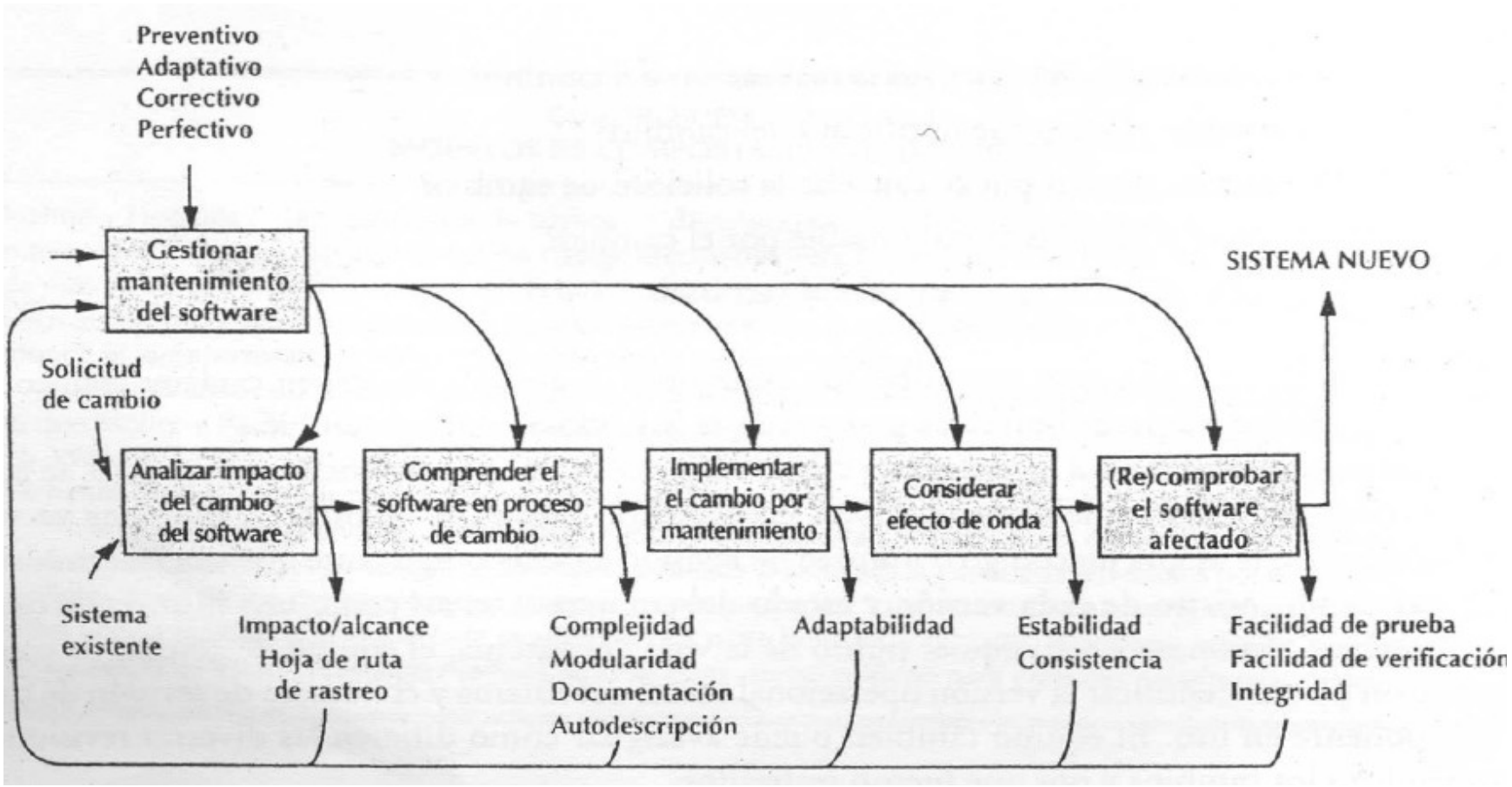
»Tipos de Mantenimiento que se realiza en un software



Fuente: Pfleeger Cap. 11

Mantenimiento

» Actividades



Fuente:

Herramientas para realizar mantenimiento

- Editores de texto
- Comparadores de archivos (https://winmerge.org/)
- Compiladores y linkeadores
- Debbuggers
- Analizadores de código estático (https://kinsta.com/es/blog/herramientas-de-revision-de-codigo/)
- Repositorios de gestión de configuración (Git, Docker, Terraform, Ansible, SaltStack, Chef, Puppet)

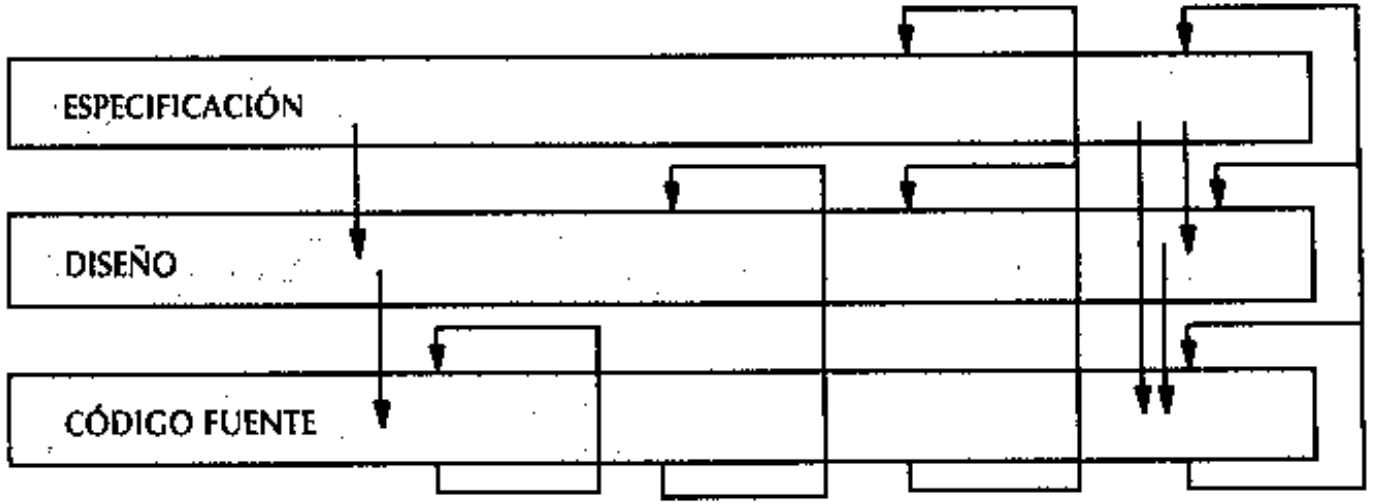
Fuente:

Rejuvenecimiento del Software

- Es un desafío del mantenimiento, intentando aumentar la calidad global de un sistema existente
- Contempla retrospectivamente los subproductos de un sistema para intentar derivar la información adicional o reformarlo de un modo comprensible
- Tipos de Rejuvenecimiento
  - Re-documentación
  - Re-estructuración
  - Ingeniería Inversa
  - Re-ingeniería

Fuente: Pfleeger Cap. 11

Rejuvenecimiento del Software



Ingeniería progresiva	*Reestructuración*	*Reestructuración de documentos*	*Ingeniería inversa*	*Reingeniería*
• avanza a través del proceso	• desde el código • representa internamente • simplifica iterativamente la estructura y elimina código muerto • regenera el código	• desde el código • informa el análisis estático sobre estructura, complejidad, volumen, datos, etc. • no está basado en métodos de software	• desde el código • produce especificación y diseño basados en métodos aceptados del software • gestiona la representación	• desde el código • hace ingeniería reversa del código • hace ingeniería progresiva: completa y modifica la representación, regenera el código

Fuente: Pfleeger Cap. 11

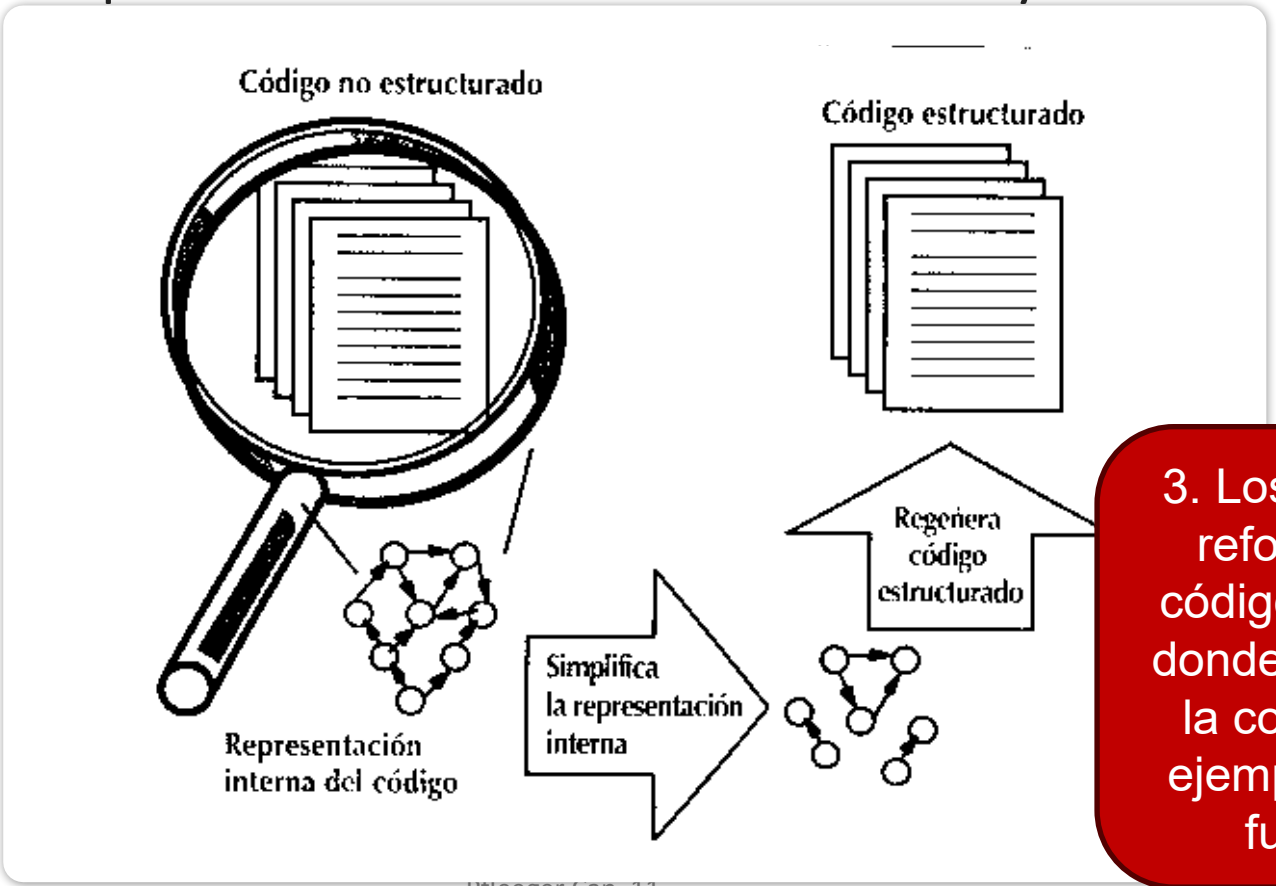
Rejuvenecimiento del Software

» **Re-estructuración**

Se reestructura el software para hacerlo más fácil de entender y de cambiar

1. Interpretamos el código fuente y lo representamos internamente como una red semántica o grafo.

2. Se utilizan reglas de transformación para simplificar la representación interna.



3. Los resultados se reformulan como código estructurado, donde se espera que la complejidad por ejemplo ciclomática fue reducida

Rejuvenecimiento del Software

Re-documentación: Representa un análisis estático del código para producir la documentación del sistema.

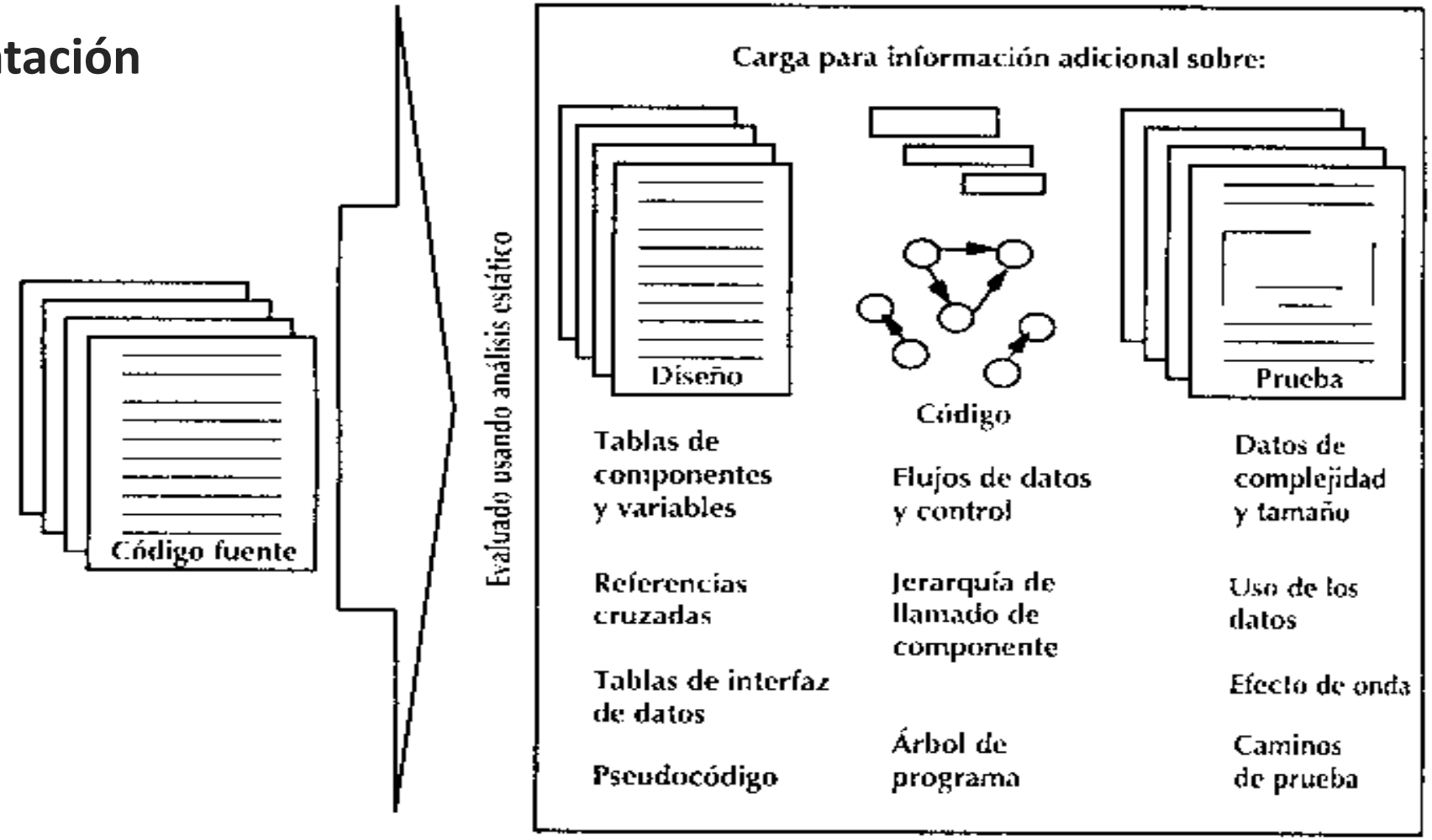
- La información producida por un análisis de código estático puede ser gráfico o textual.
- Podemos obtener: relaciones de llamada de componentes, jerarquías de clases, tablas de interfaz de datos, información del diccionario de datos, tablas o diagramas de flujo de datos, tablas o diagramas de flujo de control, pseudocódigo, caminos de prueba, referencias cruzadas de componentes y variables.

Fuente:

Pfleeger Cap. 11

Rejuvenecimiento del Software

- Re-documentación



Rejuvenecimiento del Software

- **Ingeniería Inversa**

  La ingeniería inversa es el proceso de analizar un producto o sistema para comprender su diseño, funcionamiento interno y funcionalidad.

  Parte del código fuente y recupera el diseño y en ocasiones la especificación, para aquellos sistemas en los que no hay documentación.

  Se puede utilizar Ghidra (https://ghidra-sre.org/). Se utiliza para desensamblar, descompilar y analizar código binario. Incluye un descompilador que puede convertir el código ensamblador en un lenguaje de nivel superior, como C o Java, lo que puede facilitar la comprensión de la funcionalidad de un archivo binario

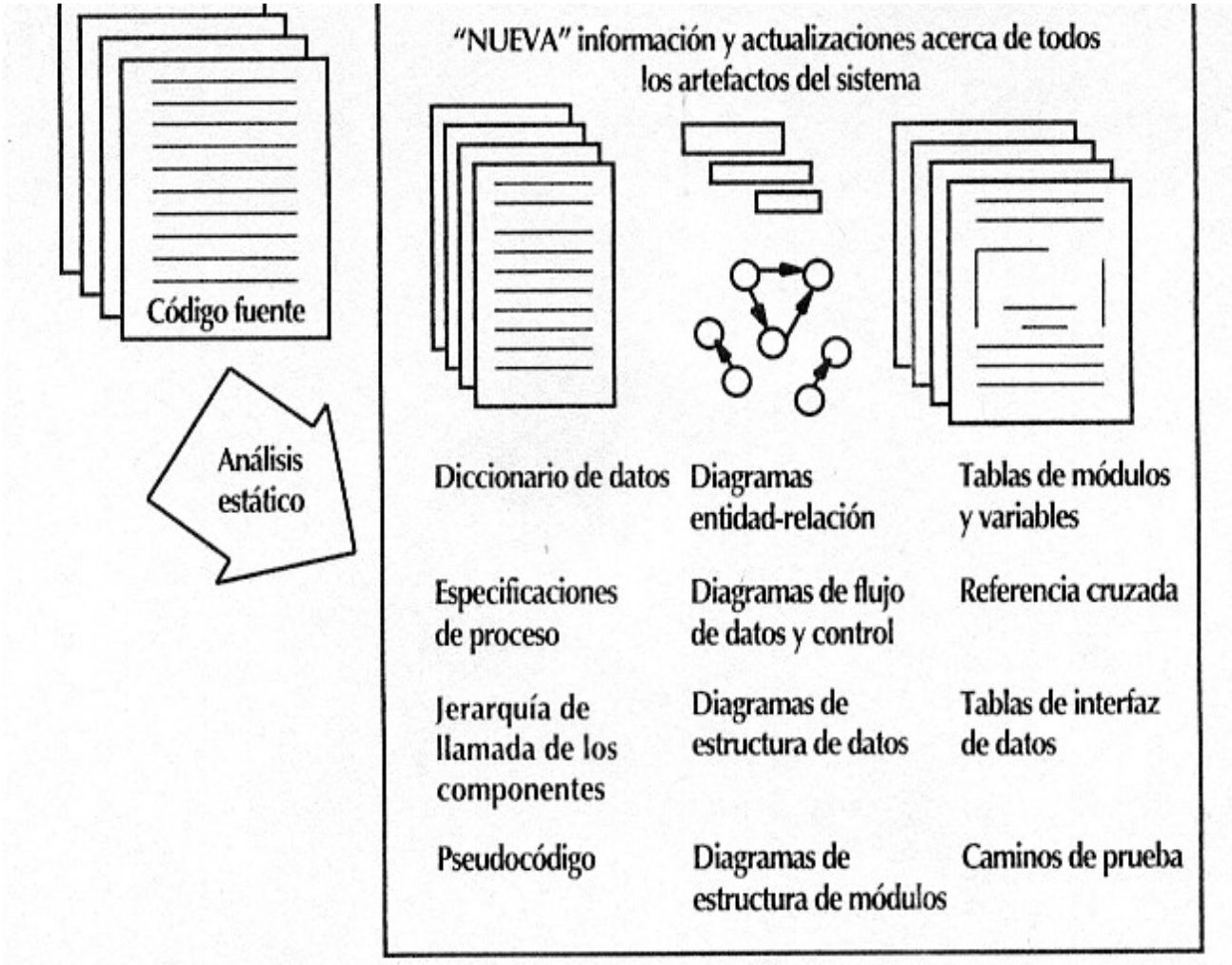
Fuente: Pfleeger Cap. 11

Rejuvenecimiento del Software

»Ingeniería Inversa

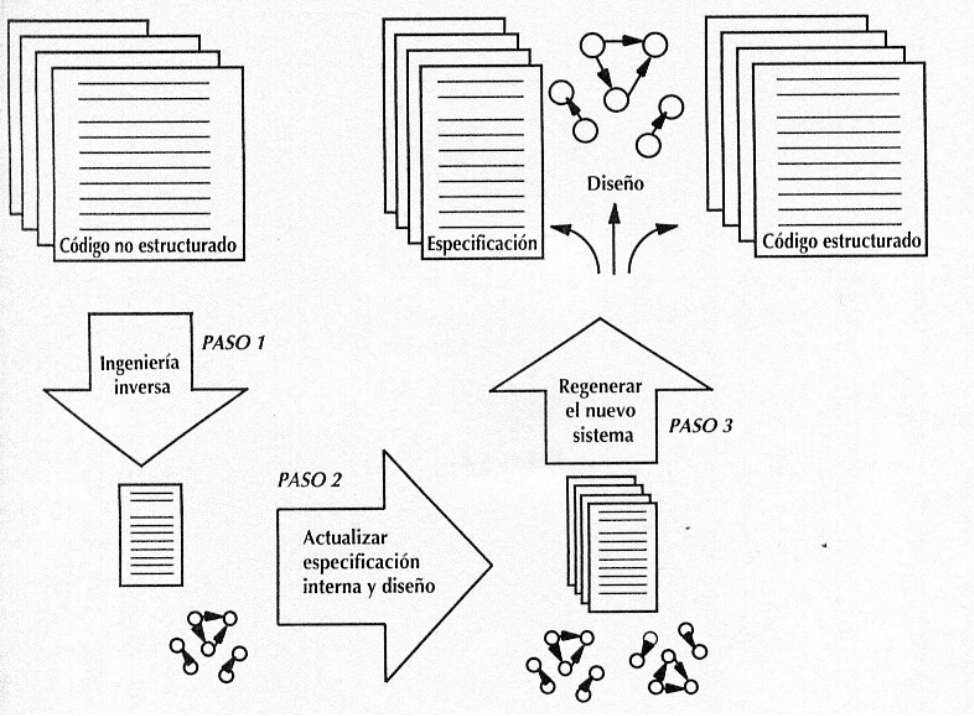
1. Se envía el código fuente. conectado a una herramienta de ingeniería inversa, que interpreta la estructura y la información de nomenclatura.

2. Se obtiene múltiples formatos hasta llegar a la especificación del sistema



Fuente:

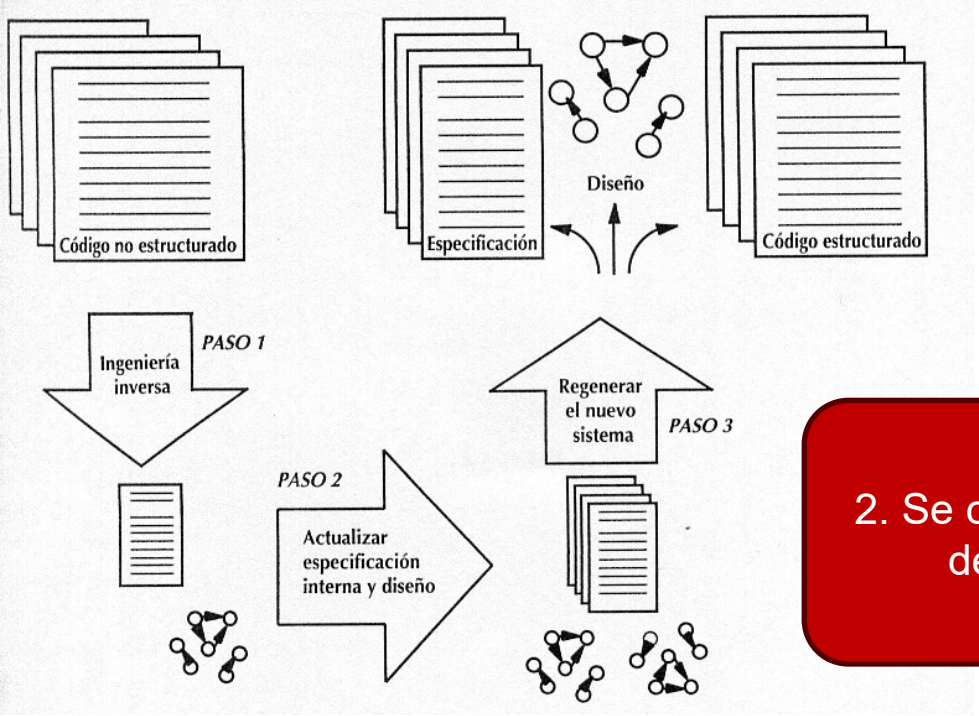
Rejuvenecimiento del Software



- **Re-ingeniería**
  - Extensión de la ingeniería Inversa
  - Produce un nuevo código fuente correctamente estructurado, mejorando la calidad sin cambiar la funcionalidad del sistema

Fuente: Pfleeger Cap. 11

Rejuvenecimiento del Software



1. El sistema tiene ingeniería inversa y se representa internamente para humanos y modificaciones de código basadas en métodos actuales para especificar y diseñar software.

2. Se corrige o completa el modelo del sistema de software.

3. El nuevo sistema se genera a partir de esta nueva especificación o diseño.

Fuente: Pfleeger Cap. 11

Futuro del rejuvenecimiento del software

- Las herramientas comerciales de ingeniería inversa recuperan parcialmente un sistema de software pueden identificar, presentar y analizar información del código fuente, pero no reconstruyen, capturan ni expresan abstracciones de diseño que no son representado explícitamente en el código fuente.
- La información del código fuente no contiene mucha información sobre el contexto original.

Fuente:

Futuro del rejuvenecimiento del software

- En general para una buena recuperación se requiere información del código, documentación de diseño existente, experiencia personal y conocimiento general sobre el dominio del problema. Conocimiento lingüístico informal sobre el dominio del problema y se necesitan modismos de aplicación antes de que un diseño completo pueda ser entendido, reconstruido y estructurado.

- El rejuvenecimiento del software avanzará cuando la tecnología y los métodos puedan capturar reglas, políticas, decisiones de diseño, terminología, convenciones de nomenclatura y otros elementos de información informal.

Fuente: