Tema 16 – Reingeniería

Ingeniería del Software

Rubén Fuentes Fernández
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Trabajando con Antonio Navarro, Juan Pavón y Pablo Gervás

Contenidos

- Sistemas heredados
- Reingeniería
  - Transformación de la arquitectura
  - Traducción de código fuente
  - Ingeniería inversa
  - Reestructuración de programas
  - Modularización de programas
  - Reingeniería de datos

Sistemas heredados

- Los *sistemas heredados* son sistemas usados por la organización para sus negocios.
  - Deben mantenerse → reingeniería
- Incluyen millones de líneas de código fuente.
  - Muchas veces escritas en lenguajes antiguos como COBOL o FORTRAN.
- Frecuentemente desarrollados antes de que el uso de las técnicas de ingeniería de software estuviera difundido.
  - No están estructurados ni documentados.
- Contienen conocimiento crítico del negocio que puede no estar documentado en ningún otro lugar.
  - No hay especificación del sistema.
- Problemas:
  - Típicamente sus costes de mantenimiento se incrementan.
  - El riesgo de reimplementar estos sistemas es muy alto.

TÉCNICAS DE REINGENIERÍA

TRANSFORMACIÓN DE LA ARQUITECTURA

Transformación de la arquitectura

- La *transformación de la arquitectura* se suele referir a la transformación desde una arquitectura centralizada a otra distribuida.
  - Aunque no todos los sistemas heredados tienen la misma arquitectura.
  - Por ello los cambios pueden afectar a distintos aspectos.
- Razones para la transformación:
  - coste de hardware → servidor es más barato
  - interfaces → los usuarios esperan GUI
  - acceso distribuido → desde distintas localizaciones geográficas

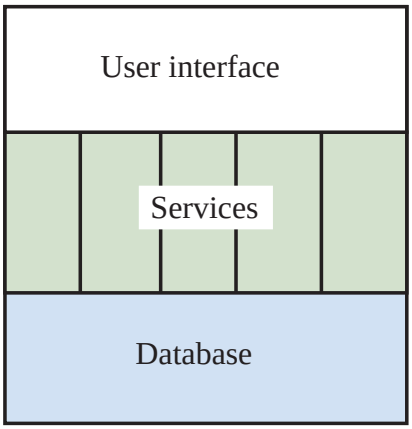
Factores para decidir

Factor	Descripción
Importancia para el negocio	El retorno de la inversión de distribuir un sistema heredado depende de su importancia para el negocio y de cuánto tiempo seguirá siéndolo. Si la distribución proporciona un soporte más eficiente para procesos de negocio estables, entonces es más probable que constituya una estrategia de evolución efectiva en coste.
Edad del sistema	Cuanto más antiguo sea el sistema, más difícil será modificar su estructura porque los cambios previos la habrán degradado.
Estructura del sistema	Cuanto más modular sea el sistema, más fácil será cambiar su arquitectura. Si la lógica de la aplicación, la gestión de los datos y la interfaz de usuario del sistema están estrechamente entrelazadas, será difícil separar las funcionalidades en la migración.
Política de adquisición de hardware	La distribución de la aplicación puede ser necesaria si la política de la compañía es reemplazar caros computadores *mainframe* por servidores más baratos.

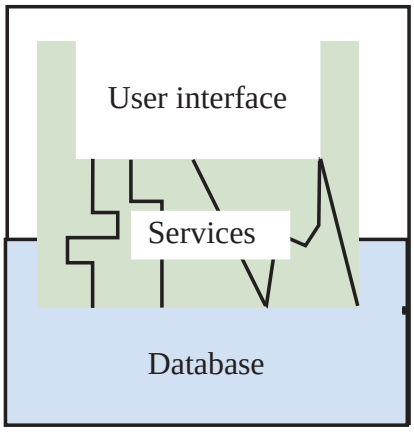
Estructura de sistemas heredados

- En arquitecturas distribuidas, debe haber una separación clara entre:
  - interfaz de usuario
  - servicios que proporciona el sistema
  - gestión de datos del sistema
- El típico ejemplo de estas arquitecturas es la arquitectura multicapa.
- La mayor parte de los sistemas heredados no siguen estas arquitecturas, por lo que en la práctica todo esto está mezclado.

Estructura en sistemas heredados

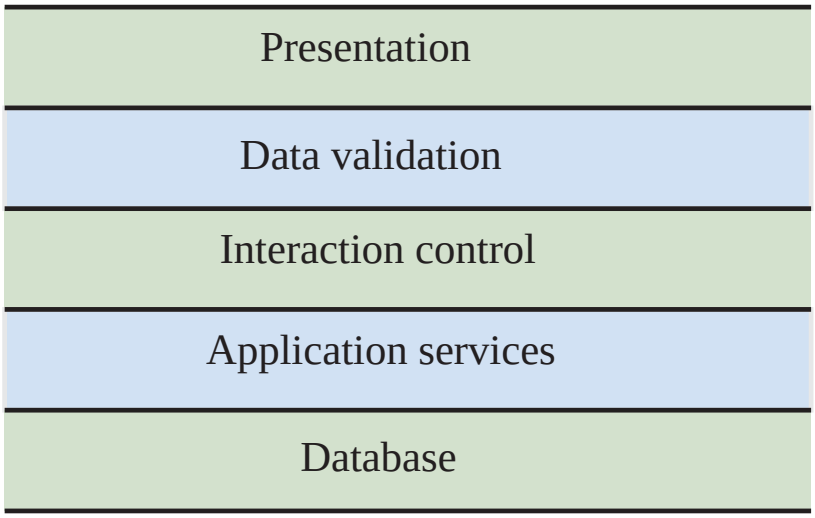


Ideal model for distribution

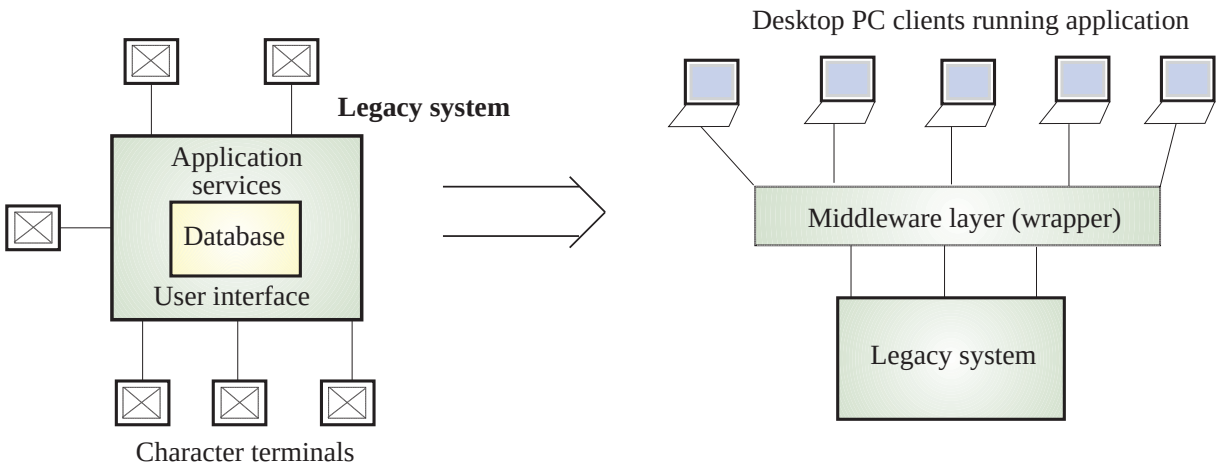


Real legacy systems

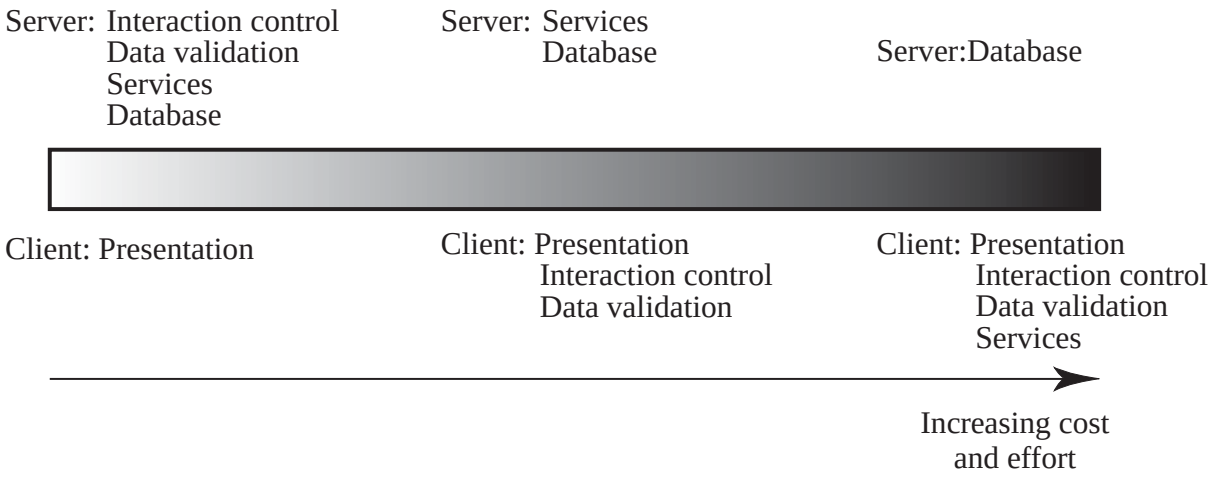
Distribución por capas



Distribución de sistemas heredados



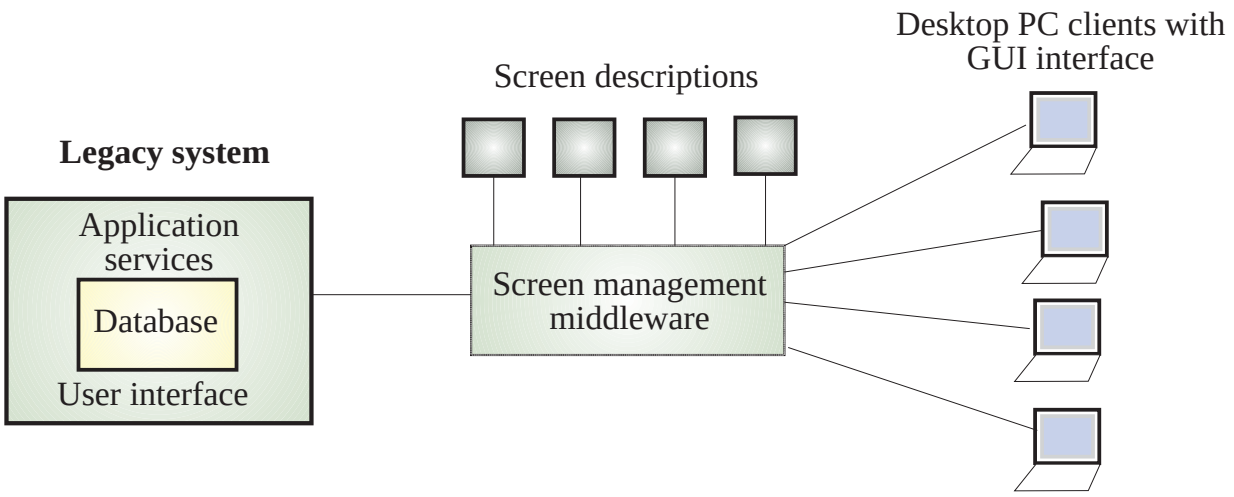
Espectro de opciones de distribución



Distribución de interfaces de usuario

- Al distribuir las interfaces de usuario, se aprovecha la capacidad del cliente para gestionar la interfaz gráfica.
- Si las partes de aplicación e interfaz están separadas, se puede convertir el sistema heredado a interfaces gráficas de usuario con cierta facilidad.
- Si no están separadas, se utiliza una aplicación intermedia para convertir las interfaces en texto a interfaces gráficas.

Conversión a GUI



Estrategias de migración

Estrategia	Ventajas	Desventajas
Implementación usando el sistema gestor de ventanas	• Acceso a toda la funcionalidad de la interfaz de usuario de forma que no hay restricciones reales en el diseño de dicha interfaz • Mejor rendimiento de la interfaz de usuario	• Depende de la plataforma • Puede ser más difícil lograr la consistencia de la interfaz de usuario
Implementación usando un navegador web	• Independiente de plataforma • Menores costes de entrenamiento debido a la familiaridad de los usuarios con la navegación web • Mayor facilidad para lograr la consistencia de la interfaz de usuario	• Potencialmente, rendimiento más pobre de la interfaz de usuario • El diseño de la interfaz de usuario está restringido por las facilidades de los navegadores web

REINGENIERÍA

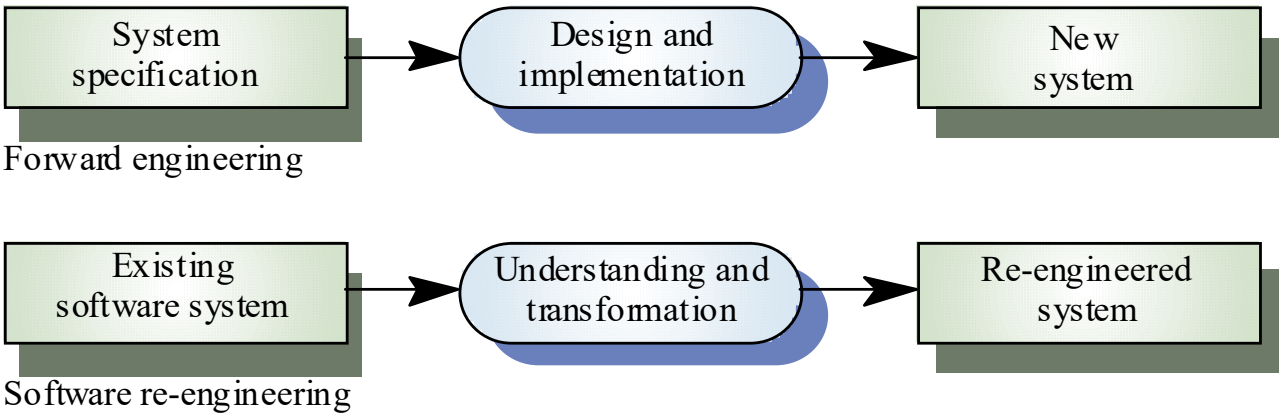
Reingeniería de sistemas

- La *reingeniería de sistemas* consiste en reestructurar o reescribir parte o todo el sistema legado sin cambiar su funcionalidad.
  - Es aplicable donde alguno (no todos) de los subsistemas de una aplicación requiere mantenimiento frecuente.
- La reingeniería incluye esfuerzo adicional para hacer más fácil el mantenimiento.
  - El sistema puede ser reestructurado y redocumentado.

Cuando hacer reingeniería

- Cuando los cambios del sistema son necesarios en sólo una parte del mismo.
- Cuando el soporte de hardware o software se hace obsoleto.
- Cuando las herramientas para soportar la reestructuración están disponibles.

Reingeniería y desarrollo

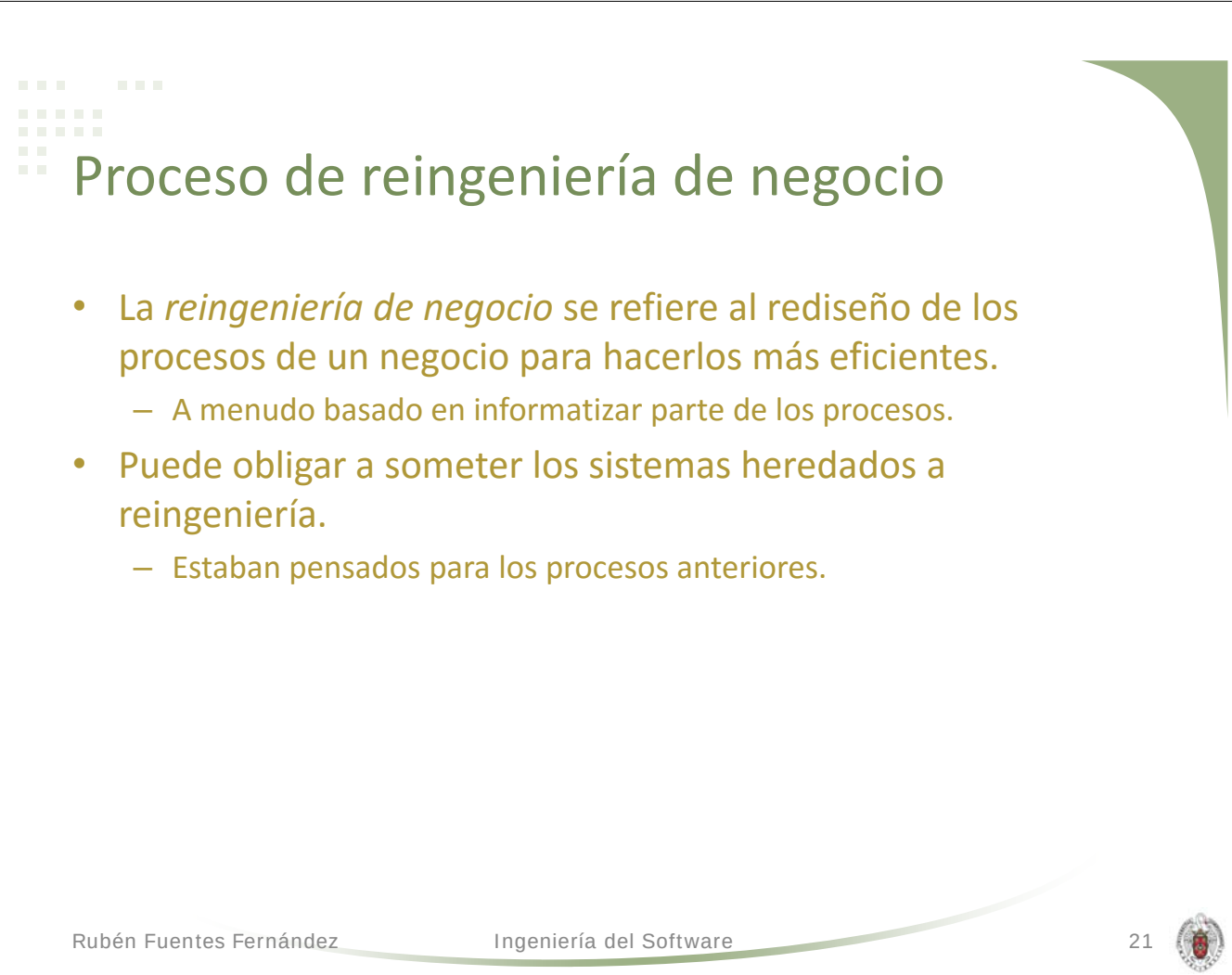


Planificación de la evolución – ruta

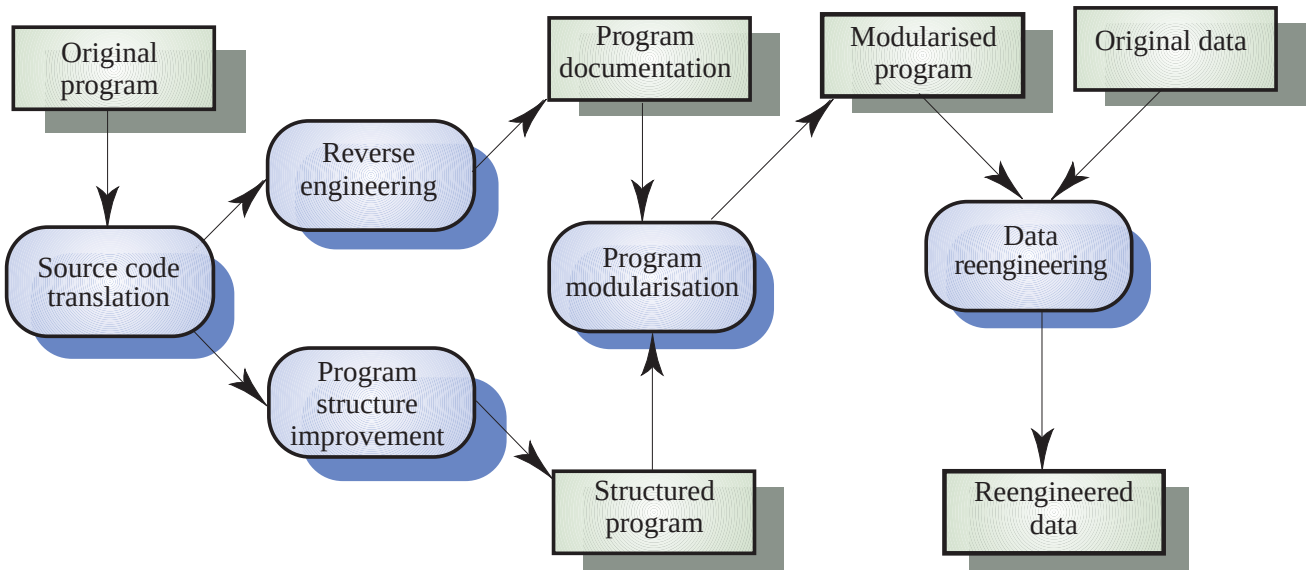
Etapa	Planificación del viaje	Planificación de la evolución
1	Determinar la razón del viaje	Establecer las restricciones a la evolución del proyecto
2	Desplegar el mapa	Definir los objetivos y procesos del negocio
3	Localizar el punto de partida	Comprender el sistema heredado
4	Localizar el destino	Formular posibles sistemas objetivo
5	Identificar posibles rutas	Formular posibles estrategias de evolución
6	Evaluar posibles rutas	Evaluar las alternativas de sistema objetivo y estrategias
7	Elegir ruta	Elegir la mejor alternativa
8	Planear el viaje	Planear la evolución del proyecto

Proceso de reingeniería de negocio

- La *reingeniería de negocio* se refiere al rediseño de los procesos de un negocio para hacerlos más eficientes.
  - A menudo basado en informatizar parte de los procesos.
- Puede obligar a someter los sistemas heredados a reingeniería.
  - Estaban pensados para los procesos anteriores.



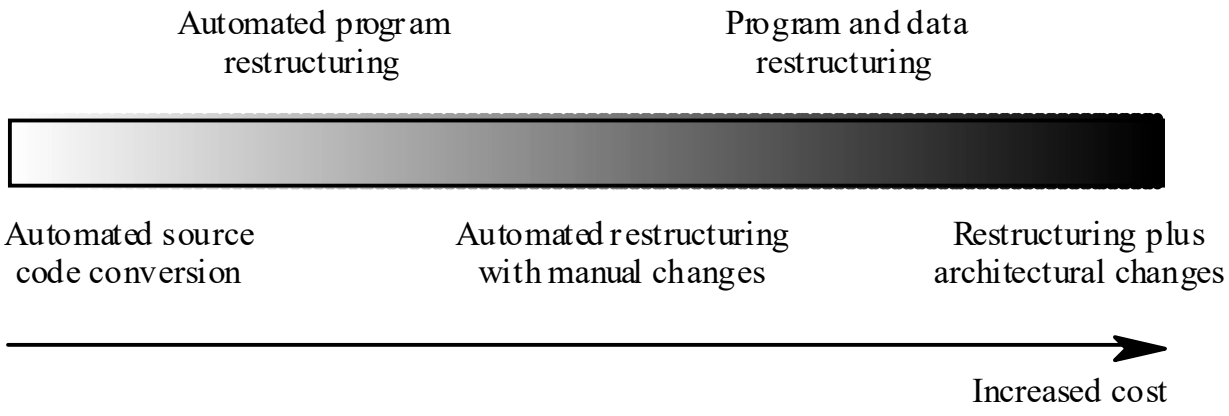
Reingeniería de programas



Reingeniería: factores de coste

- La calidad del software que será objeto de reingeniería
- Las herramientas disponibles para reingeniería
- El alcance necesario en la conversión de datos
- La disponibilidad de personal experto para hacer reingeniería

Puntos de vista en la reingeniería



Ventajas de la reingeniería

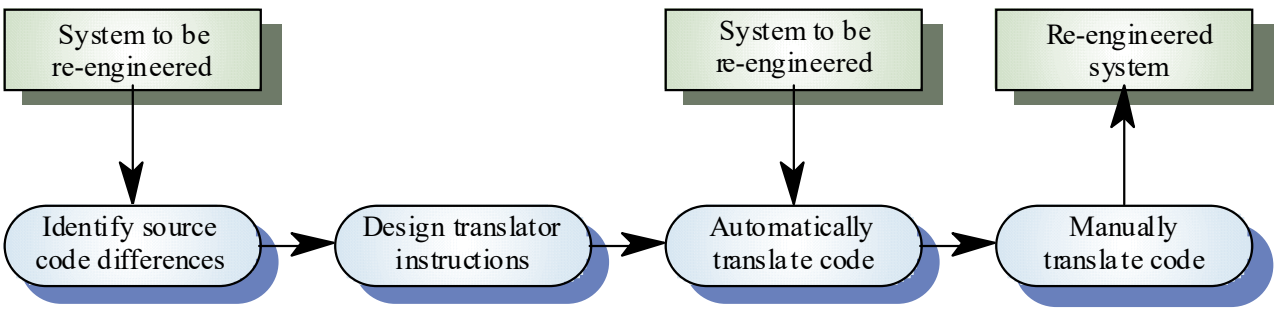
- Reduce el riesgo
  - El redesarrollo de software esencial para una empresa es una actividad de alto riesgo debido al papel crítico del software en el negocio.
- Reduce costes
  - Algunas medidas sugieren que el coste de la reingeniería es significativamente menor que el coste del redesarrollo.
    - Quizá hasta cuatro veces menos.

TRADUCCIÓN DE CÓDIGO FUENTE

Traducción de código fuente

- La *traducción de código fuente* implica convertir el código de un lenguaje (o de una versión de lenguaje) a otro.
  - Ej. de FORTRAN a C
- Puede ser necesario debido a:
  - Actualización de la plataforma de hardware
  - Falta de conocimiento del personal
  - Cambio en las políticas de la empresa
- Sólo es factible si hay un traductor automático disponible.
  - O el esfuerzo manual necesario es reducido.

El proceso de traducción de programas

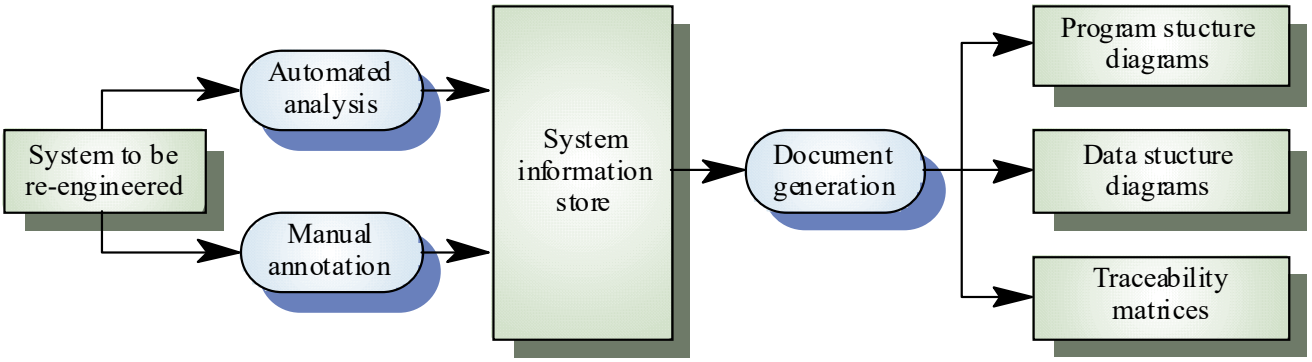


INGENIERÍA INVERSA

Ingeniería inversa

- La *ingeniería inversa* consiste en analizar software para entender su diseño y especificación.
- Puede ser parte de distintos procesos:
  - Reingeniería
  - Reespecificación de un sistema para su reimplantación
  - Generación de información tras construir un sistema
  - Obtener un diseño y especificación de partida para el sistema que ha de sustituir a otro
  - Obtener un diseño y especificación como apoyo para los procesos de mantenimiento
- Se pueden usar algunas herramientas en este proceso.
  - Ej. navegadores de especificaciones y generadores de referencias cruzadas

El proceso de ingeniería inversa



REESTRUCTURACIÓN DE PROGRAMAS

Reestructuración de programas

- La *reestructuración de programas* son cambios de los mismos para mantener una estructura interna del código simple y coherente.
- El mantenimiento tiende a corromper la estructura de un programa.
  - Cada vez es más difícil de entender.
  - Hay que actuar para no tener un código imposible de mantener.
- La reestructuración puede mejorar esta situación con varias acciones:
  - Quitar automáticamente ramificaciones incondicionales.
  - Simplificar condiciones para hacerlas más legibles.
  - …

Lógica spaghetti

```
Start:    Get (Time-on, Time-off, Time, Setting, Temp, Switch)
          if Switch = off goto off
          if Switch = on goto on
          goto Cntrld
off:  if Heating-status = on goto Sw-off
          goto loop
on:  if Heating-status = off goto Sw-on
          goto loop
Cntrld:   if Time = Time-on goto on
          if Time = Time-off goto off
          if Time < Time-on goto Start
          if Time > Time-off goto Start
          if Temp > Setting then goto off
          if Temp < Setting then goto on
Sw-off:   Heating-status := off
          goto Switch
Sw-on:    Heating-status := on
Switch:   Switch-heating
loop:     goto Start
```


Lógica de control estructurada

```
loop
      -- The Get statement finds values for the given variables from the system's
-- environment.
      Get (Time-on, Time-off, Time, Setting, Temp, Switch) ;
      case Switch of
            when On => if Heating-status = off then
                              Switch-heating ; Heating-status := on ;
                        end if ;
            when Off => if Heating-status = on then
                              Switch-heating ; Heating-status := off ;
                        end if;
            when Controlled =>
                  if Time >= Time-on and Time < = Time-off then
                        if Temp > Setting and Heating-status = on then
                              Switch-heating; Heating-status = off;
                        elsif Temp < Setting and Heating-status = off then
                              Switch-heating; Heating-status := on ;
                        end if;
                  end if ;
      end case ;
end loop ;
```


Simplificación de una condición

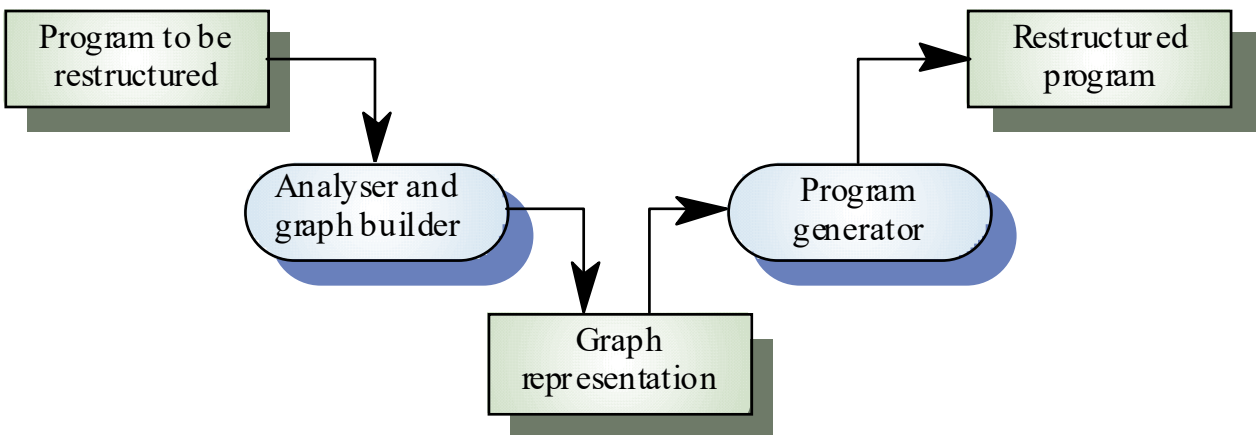
```
-- condición compleja
If  not (A > B and (C < D or not (E > F) ) ) ...

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


Problemas de la reestructuración

- Los problemas con la reestructuración son:
  - Pérdida de comentarios
  - Pérdida de documentación
  - Fuertes demandas de cómputo
- La reestructuración no resuelve los problemas de modularización si los componentes relacionados están dispersos.
- La comprensión de programas que manejan datos puede no mejorarse con la reestructuración.

Programa de reestructuración automática



MODULARIZACIÓN DE PROGRAMAS

Modularización de programas

- La *modularización de programas* es reorganizar un programa para que las partes relacionadas se agrupen en un sólo módulo.
- Normalmente es un proceso manual basado en inspección y reorganización del código.

Tipos de módulo

- Abstracciones de datos
  - Tipos abstractos de datos y similares
- Módulos de hardware
  - Funciones para el manejo de un hardware concreto
- Funcionales
  - Agrupan funciones relacionadas entre sí
- De proceso
  - Funciones relacionadas con un proceso concreto

Recuperando abstracciones de datos

- Muchos sistemas heredados usan tablas compartidas para ahorrar memoria.
  - Causan problemas debido a que los cambios tienen un alto impacto en el sistema.
- Los datos globales pueden convertirse a objetos o tipos abstractos de datos.
  - Analizar áreas comunes de datos para identificar abstracciones lógicas.
  - Crear un tipo abstracto de datos u objeto para esas abstracciones.
  - Encontrar todas las referencias a datos y remplazarlas con una referencia a una abstracción de datos.

REINGENIERÍA DE DATOS

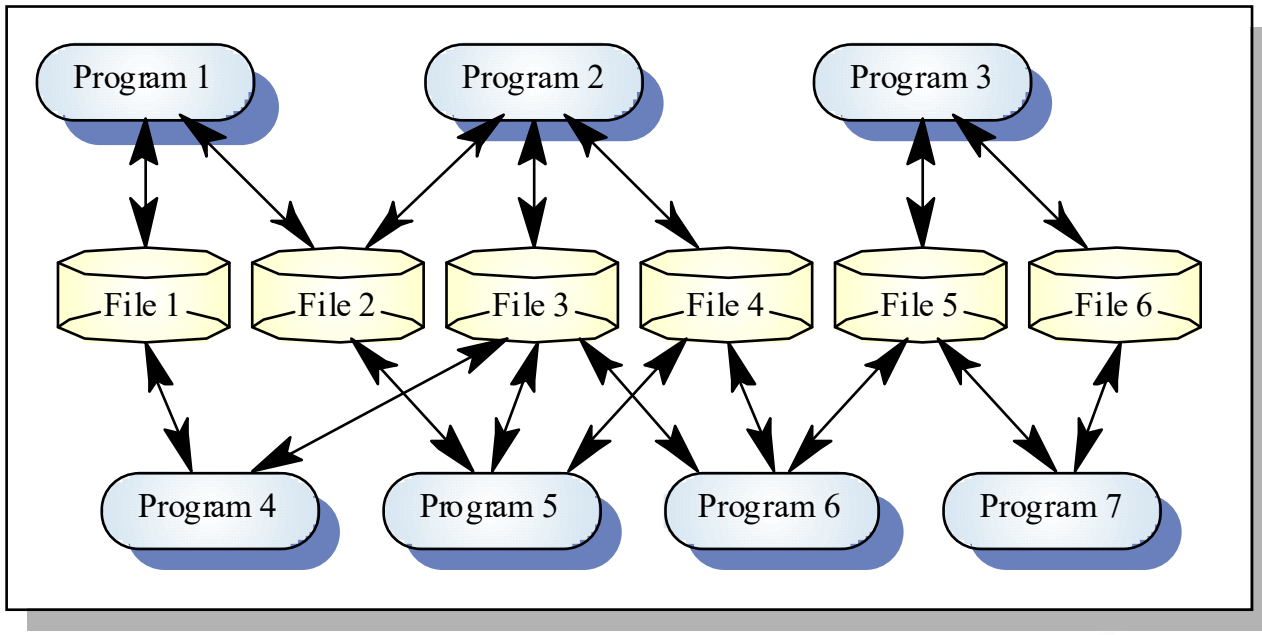
Reingeniería de datos

- La *reingeniería de datos* incluye el análisis y la reorganización de las estructuras de datos (y algunas veces los valores de los datos) en un programa.
- Puede ser parte de distintos procesos de migración:
  - De un sistema de archivos a un Sistema Gestor de Bases de Datos (*DataBase Management System*, DBMS)
  - Del cambio de un DBMS a otro o entre versiones del mismo
- El objetivo es crear un medio administrable de datos.

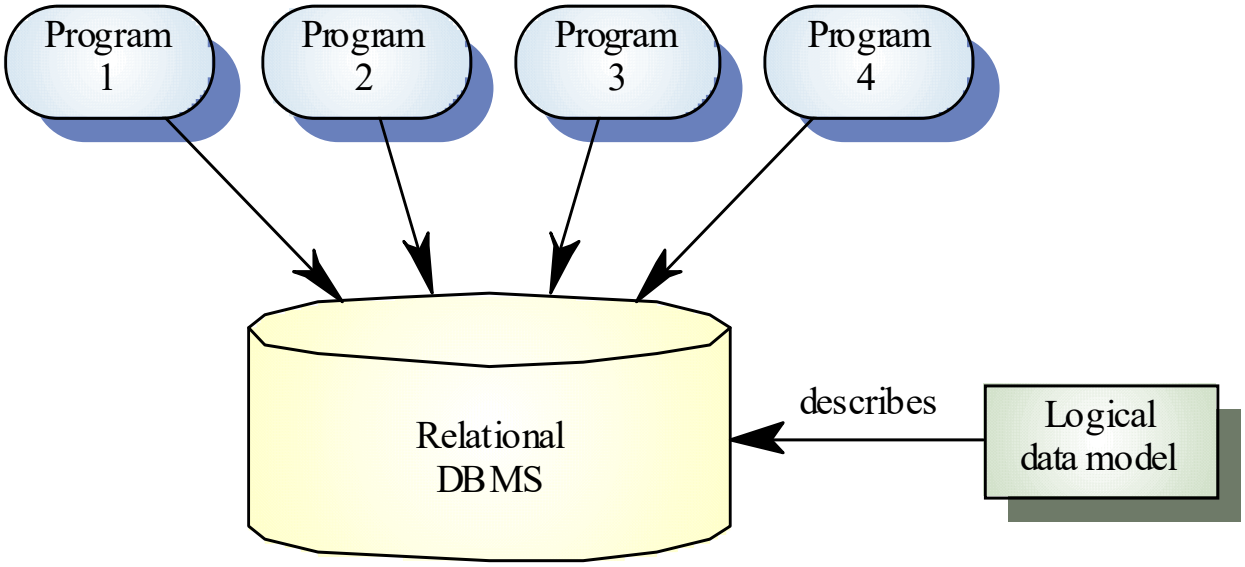
Aproximaciones a la reingeniería de datos

Aproximación	Descripción
Limpieza de datos	Se analizan los registros y valores de datos para mejorar su calidad. Se elimina la información redundante y los duplicados, y se aplica un formato consistente a todos los registros. Este proceso no requiere generalmente cambios en el programa asociado.
Extensión de datos	Los datos y el programa asociado son sometidos a reingeniería para eliminar límites en el procesamiento. Esto puede requerir cambios en el programa para incrementar las longitudes de campos, modificar los límites superiores de tablas… Los datos pueden ser entonces reescritos y limpiados para reflejar los cambios en el programa.
Migración de datos	Los datos se migran para ser gestionados mediante un DBMS. Los datos pueden estar almacenados en ficheros o ser gestionados por un DBMS más antiguo.

Distribución de datos en sistemas heredados



Entorno de administración de datos



Problemas con los datos (1/2)

- Los usuarios finales quieren datos en sus máquinas en lugar de sistemas de archivos.
  - Necesitan descargar estos datos de un 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.

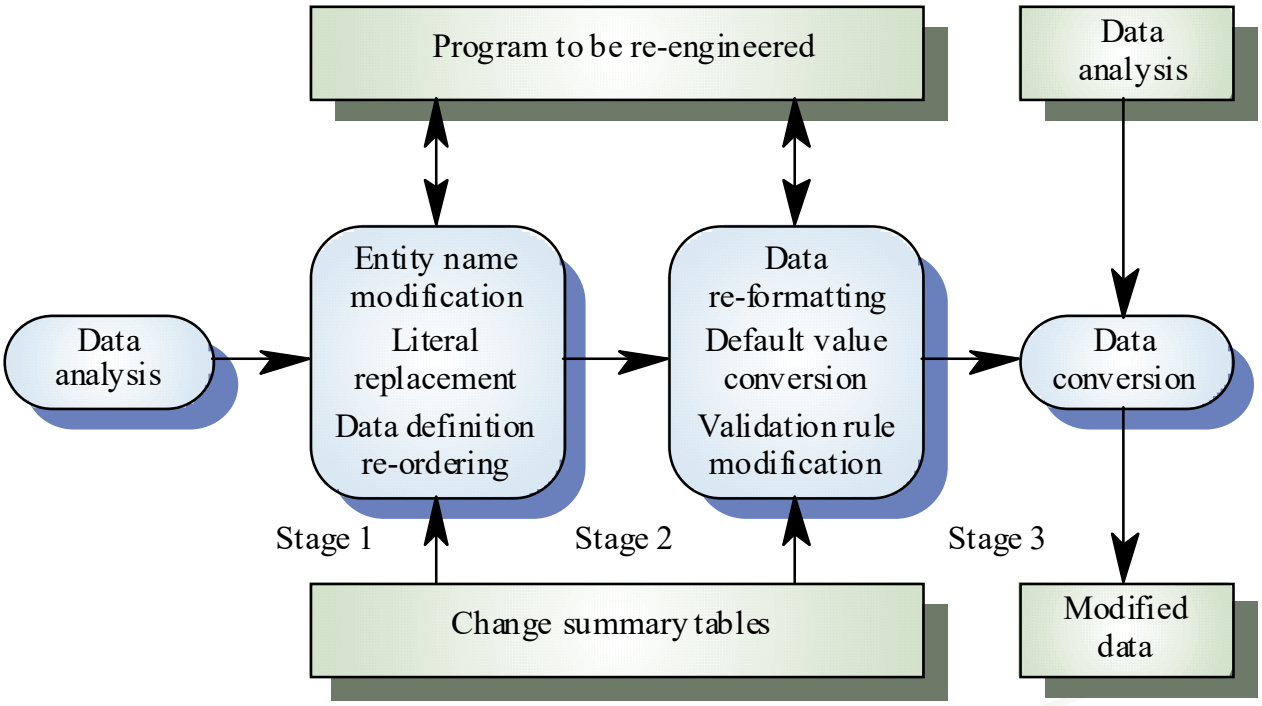
Problemas con los datos (2/2)

- Problemas al nombrar los datos
  - Los nombres pueden ser difíciles de entender.
  - Los mismos datos pueden tener diferentes nombres en diferentes programas.
- Problemas con la longitud de los campos
  - El mismo elemento puede asignarse a diferentes longitudes en diferentes programas.
- Problemas con la organización de registros
  - Registros representando la misma entidad pueden estar organizados de diferente manera en diferentes programas.
- Literales difíciles de codificar
- No hay diccionario de datos

Inconsistencias en los valores de los datos

Inconsistencia de los datos	Descripción
Inconsistencia de los valores por defecto	Diferentes programas asignan diferentes valores por defecto. Esto causa problemas en programas diferentes de los que crearon los datos. El problema se agudiza cuando se asignan valores por defecto válidos a los datos perdidos, puesto que entonces no se pueden identificar los datos no disponibles.
Unidades inconsistentes	La misma información se representa con diferentes unidades en distintos programas.
Reglas de validación inconsistentes	Diferentes programas aplican diferentes reglas de validación a los datos. Los datos aceptados por un programa pueden ser rechazados por otro. Este problema está muy presente con los datos históricos, que pueden no haber sido actualizados en línea con las nuevas reglas de validación.
Semántica de la representación inconsistente	El programa asume cierto significado de la forma en la que los datos están representados. Los programas pueden utilizar diferentes convenciones de representación.
Gestión inconsistente de valores negativos	Algunos programas rechazan valores negativos para datos que siempre deben ser positivos. Otros los aceptan o fallan al reconocerlos como negativos, y los convierten a positivos.

El proceso de reingeniería de datos



Conversión de datos

- La reingeniería de datos puede incluir el cambio de la organización de la estructura de datos sin cambiar los valores de los datos.
- La conversión de valores de datos puede ser costosa.
  - Se tendrá que escribir algún programa o *script* con este propósito.

CONCLUSIONES

Conclusiones

- Los procesos de reingeniería forman parte de las actividades habituales de la Ingeniería del Software.
  - Constituyen una parte muy importante del negocio.
- Contemplan diferentes procesos dependiendo del contexto de aplicación y los objetivos.
- Generalmente comienzan con tareas para comprender el sistema heredado.
  - Difíciles porque estos sistemas pueden no contar con una estructura ni documentación adecuadas.
- Existen herramientas que automatizan parcialmente algunos de estos procesos.

Glosario

- DBMS = *DataBase Management System*
- GUI = *Graphical User Interface*

Referencias

- R. Pressman: Ingeniería del Software. Un enfoque práctico, 7ª edición. McGraw-Hill, 2010.
  - Capítulo 29
- I. Sommerville: Ingeniería del Software, 7ª edición. Addison Wesley, 2007.
  - Capítulo 21