Mantenimiento del Software

S6

Francisco Ruiz, Macario Polo

Grupo Alarcos

Dep. de Informática

ESCUELA SUPERIOR DE INFORMÁTICA

UNIVERSIDAD DE CASTILLA-LA MANCHA

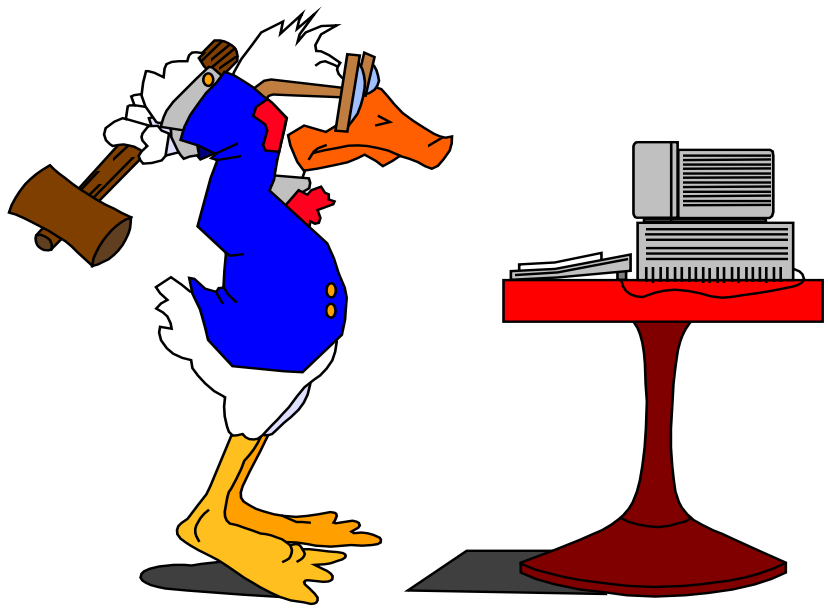
http://alarcos.inf-cr.uclm.es/doc/mso/

Ciudad Real, 2000/2001

Índice - Sesión 6

- Soluciones técnicas.
  - Reingeniería.
  - Ingeniería inversa.
  - Reestructuración.
- Ingeniería inversa de bases de datos
- Detección de clones

- Métodos estructurados
- Métodos orientados a objetos
- El ciclo de vida
- El modelo en cascada
- ...



Soluciones técnicas

- Ingeniería inversa
- Reingeniería
- Reestructuración
  - –Detección de clones

Ingeniería inversa

Análisis de un sistema, de manera que se produce una representación a alto nivel del propio sistema.

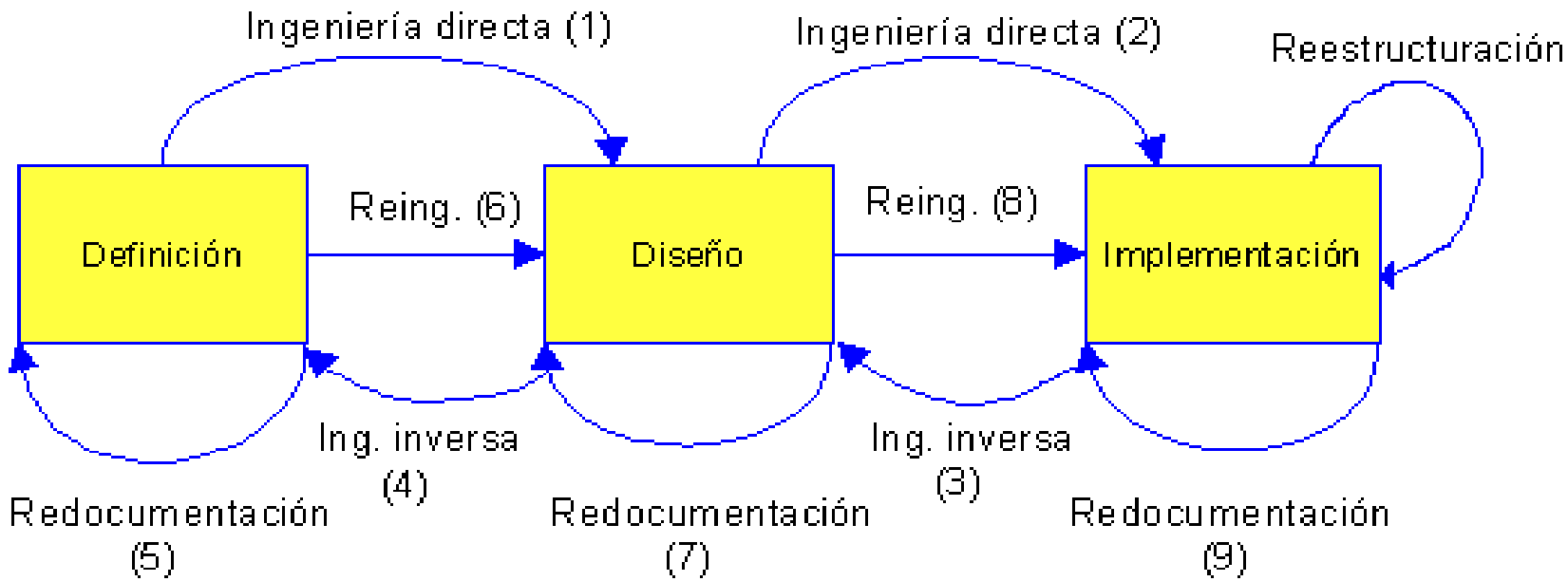
Reingeniería

Modificación de un producto software, 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, de tal manera que se oriente este cambio hacia mayores niveles de facilidad en cuanto a mantenimiento, reutilización, comprensión o evolución.

Reestructuración

Cambio de representación de un producto software, pero dentro del mismo nivel de abstracción.

Gráficamente...



Soluciones técnicas: motivaciones

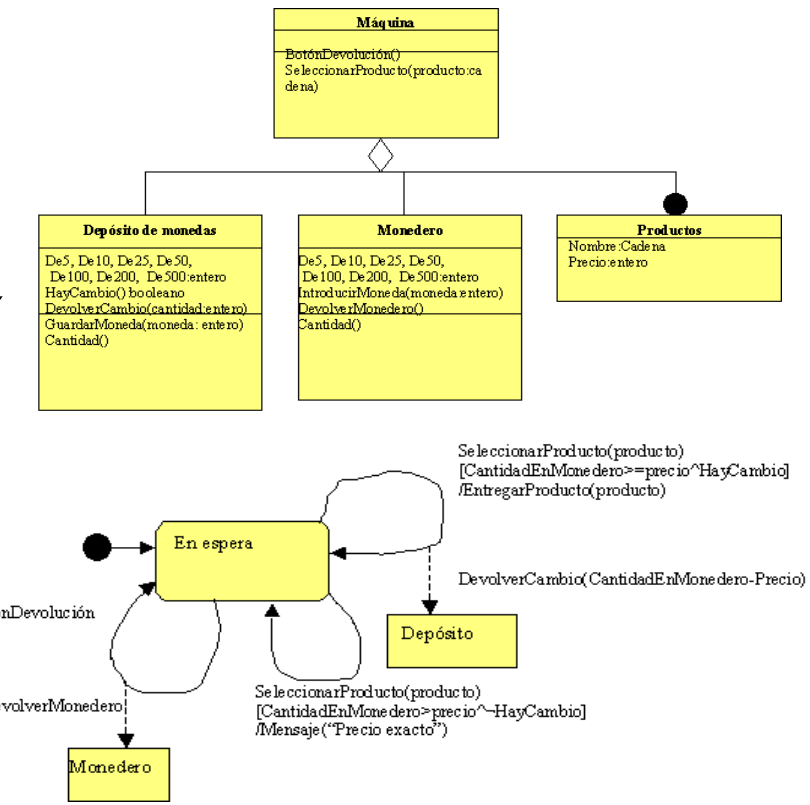
- Obtención de documentación
- Transformación de aplicaciones *procedimentales* al paradigma orientado a objeto
- Transformación de sistemas transaccionales a cliente/servidor
- Paso de un lenguaje a otro, dentro del mismo paradigma
- ...

Ingeniería inversa (I)

¿De qué?

- De procesos

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include "maquina.h"

Maquina :: Maquina(DepositoDeMonedas *deposito, Monedero *monedero,
Producto lista[3]) {
      m_deposito=deposito;
      m_monedero=monedero;
      for (int i=0; i<=3; i++)
            m_lista[i]=lista[i];
}

void Maquina :: Mostrar() {
      m_deposito->Mostrar();

      for (int i=0; i<=3; i++)
            printf("Producto %d: %s, a %d pts\n", i,
m_lista[i].nombre(), m_lista[i].precio());
      printf("\n");
}

void Maquina :: GuardarMonederoEnDeposito() {
      int i;

      for (i=0; i<=m_monedero->m_De5; i++)
            m_deposito->GuardarMoneda(5);
      m_monedero->m_De5=0;
      for (i=0; i<=m_monedero->m_De10; i++)
            m_deposito->GuardarMoneda(10);
      m_monedero->m_De10=0;
      for (i=0; i<=m_monedero->m_De25; i++)
            m_deposito->GuardarMoneda(25);
      m_monedero->m_De25=0;
      for (i=0; i<=m_monedero->m_De50; i++)
            m_deposito->GuardarMoneda(50);
      m_monedero->m_De50=0;
      for (i=0; i<=m_monedero->m_De100; i++)
            m_deposito->GuardarMoneda(100);
      m_monedero->m_De100=0;
      for (i=0; i<=m_monedero->m_De200; i++)
            m_deposito->GuardarMoneda(200);
      m_monedero->m_De200=0;
      for (i=0; i<=m_monedero->m_De500; i++)
            m_deposito->GuardarMoneda(500);
      m_monedero->m_De500=0;
}

void Maquina :: SeleccionarProducto(int producto) {
      int precio=m_lista[producto].precio();

      if (precio>m_monedero->CantidadEnMonedero()) {
            printf("Mete mas monedas: has metido %d pts y el %s vale
%d\n",
                  m_monedero->CantidadEnMonedero(),
m_lista[producto].nombre(), precio);
            m_monedero->DevolverMonedero();
      } else {
            if (m_deposito->DevolverCambio(m_monedero-
>CantidadEnMonedero()-precio, m_monedero))
```

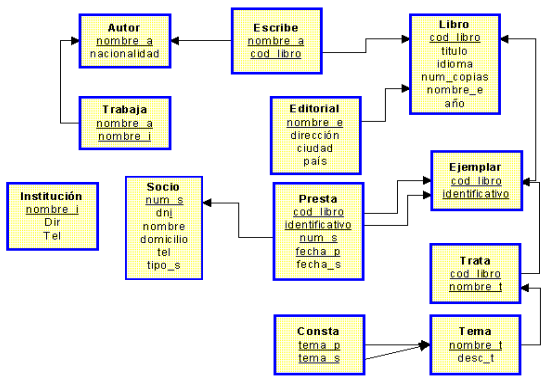


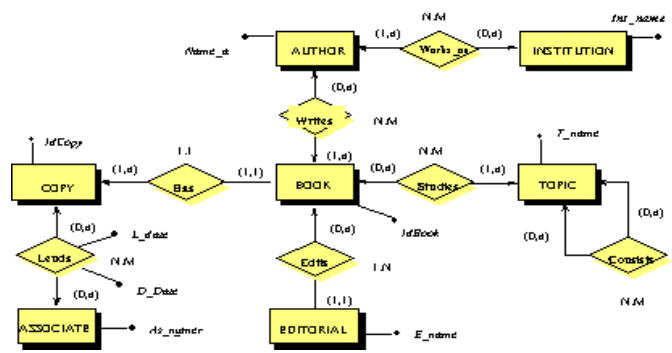
Ingeniería inversa (II)

¿Y de qué más?

- De bases de datos

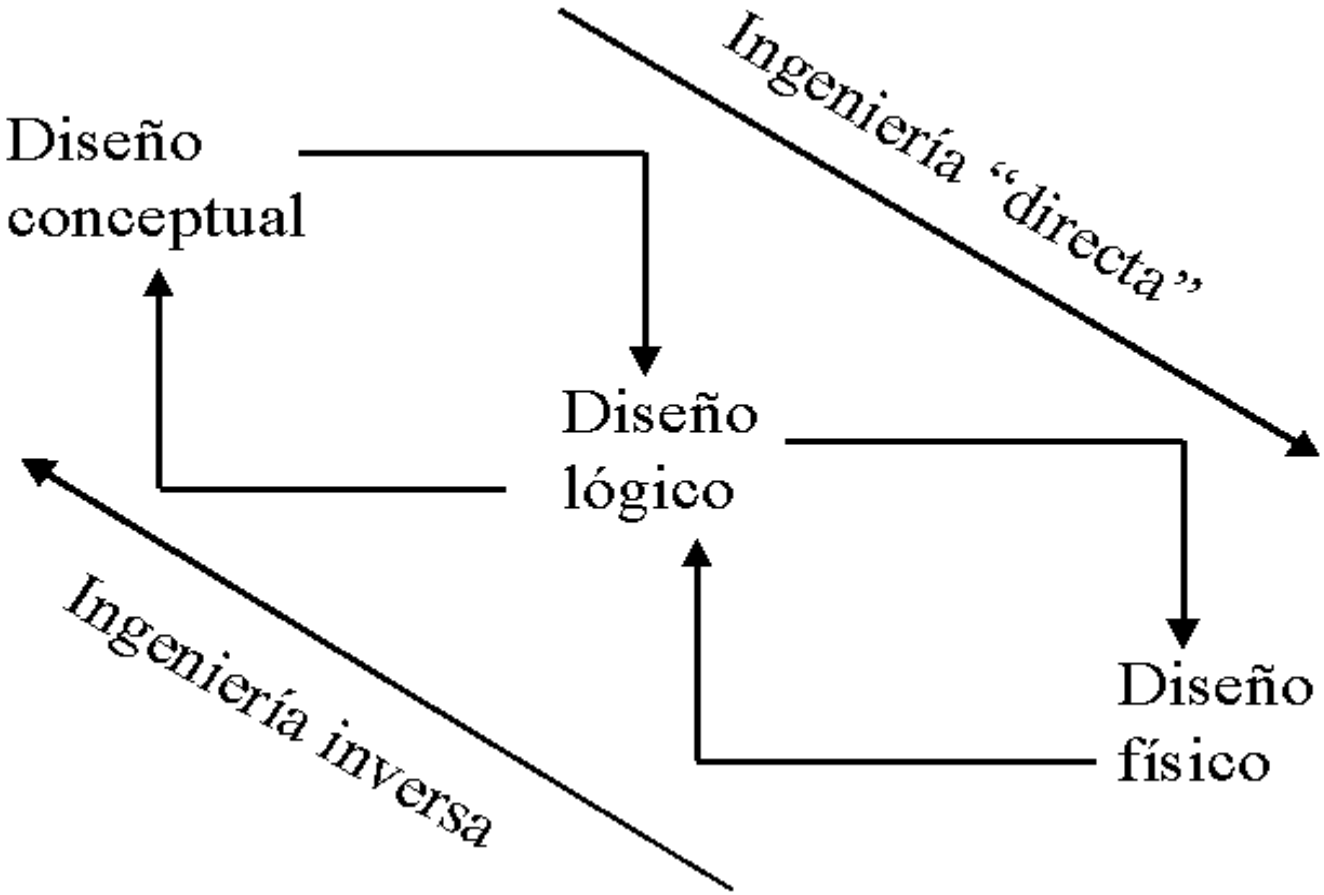
```
********** TABLAS ******************
CREATE TABLE autor
 (nombre_a          nombres,
 nacionalidad       nacionalidades,
 PRIMARY KEY (nombre_a) )

CREATE TABLE trabaja
 (nombre_a          nombres,
 nombre_i           instituciones,
 PRIMARY KEY (nombre_a, nombre_i),
 FOREIGN KEY (nombre_a) REFERENCES autor
          ON UPDATE CASCADE)

CREATE TABLE institucion
 (nombre_i          instituciones,
 Dir                lugares,
 Tel                telefonos,
 PRIMARY KEY (nombre_i))

CREATE TABLE libro
(cod_libro          códigos,
 título             título NOT NULL,
 idioma   idiomas NOT NULL,
 num_copias         número_ejemplar NOT NULL,
 nombre_e           nombres NOT NULL,
 año                año,
PRIMARY KEY (cod_libro),
FOREIGN KEY (nombre_e) REFERENCES editorial
          ON UPDATE CASCADE,
```





Ingeniería inversa de bases de datos relacionales

Ingeniería inversa de bases de datos relacionales



Ingeniería inversa de bases de datos relacionales

Comparación de métodos (Pedro de Jesús y Sousa, 1999)

- Conocimiento semántico de:
  - Atributos
  - Consistencia de nombres
- Datos:
  - Utilización de los datos
  - Existencia de errores en los datos
- Código
  - Utilización del código
  - Existencia de errores en el código
- Claves candidatas
- Claves ajenas (externas)
- Dependencias funcionales de atributos no claves o 3FN
- Dependencias de inclusión no basadas en claves
- Casos en los que se requiere entradas proporcionadas por personas

Ingeniería inversa de bases de datos relacionales

Método de Chiang et al. (1994 y 1995)

- Entradas:
  - Datos
  - Relaciones
  - Claves primarias
- Suposiciones:
  - 3FN
  - Consistencia de nombres
  - Ausencia de errores en los valores de los atributos clave
- Salidas:
  - Modelo ER extendido (EER)

Ingeniería inversa de bases de datos relacionales

Método de Johannesson (1994).

- Entradas:
  - Relaciones
  - Dependencias funcionales
  - Dependencias de inclusión
- Suposiciones:
  - 3FN
- Salidas:
  - Un par *(L, IC)*, donde *L* es un lenguaje e *IC* es un conjunto de restricciones sobre la base de datos

Ingeniería inversa de bases de datos relacionales

Método de Markowitz y Makowsk (1990).

- Entradas:
  - Relaciones
  - Dependencias clave
  - Restricciones de integridad referencial
- Suposiciones:
  - Relaciones en FN de Boyce-Codd
- Salidas:
  - Modelo ER extendido (EER)

Ingeniería inversa de bases de datos relacionales

Método de Navathe y Awong.

- Entradas:
  - Relaciones
- Suposiciones:
  - Relaciones en 3FN o FN de Boyce-Codd
  - Consistencia en los nombres de los atributos
  - Ausencia de ambigüedades u homónimos en las claves ajenas
  - Especificación de todas las claves candidatas
- Salidas:
  - Modelo ER extendido (EER)

Ingeniería inversa de bases de datos relacionales

Método de Petit et al (1996).

- Entradas:
  - Relaciones con restricciones de unicidad no nulas
  - Datos
  - Código
- Suposiciones:
  - Ninguna
- Salidas:
  - Modelo ER extendido (EER)

Ingeniería inversa de bases de datos relacionales

Método de Premerlani y Blaha (1998).

- Entradas:
  - Relaciones
  - Datos
- Suposiciones:
  - Ninguna
- Salidas:
  - Modelo de clases OMT

Ingeniería inversa de bases de datos relacionales

Método de Signore et al (1994).

- Entradas:
  - Relaciones
  - Código
- Suposiciones:
  - Ninguna
- Salidas:
  - Modelo ER

Ingeniería inversa de bases de datos relacionales

Resumen de métodos.

Método	Entradas	Salidas	Precondiciones
Chiang et al.	Datos Relaciones Claves primarias	EER	3FN Consistencia de nombres Ausencia de errores en PK
Johannesson	Relaciones Dependencias funcionales Dependencias de inclusión	Par (L, IC)	3FN
Markowitz y Makowsk	Relaciones Dependencias clave Restricciones de integridad referencial	EER	Relaciones en FN de Boyce-Codd
Navathe y Awong	Relaciones	EER	Relaciones en 3FN o FN de Boyce-Codd Consistencia en los nombres de los atributos Ausencia de ambigüedades u homónimos en FK Especificación de todas las claves candidatas
Petit et al	Relaciones con restric. de unicidad no nulas Datos Código	EER	Ninguna
Premerlani y Blaha	Relaciones Datos	Modelo de clases OMT	Ninguna
Signore et al	Relaciones Código	ER	Ninguna

Ingeniería inversa de bases de datos relacionales

Método de Hainaut et al. (I)

- Fase 1) Extracción de estructuras
  - Considerar cada fichero una posible tabla
  - Considerar campo del fichero como un posible campo de la tabla
  - Identificar un conjunto de campos susceptibles de formar la clave primaria
  - Determinar las claves externas
  - "Filtrar" las tablas (despreciar, p. ej., aquellos ficheros sin clave principal)
  - Buscar y encontrar generalizaciones
  - Encontrar asociaciones

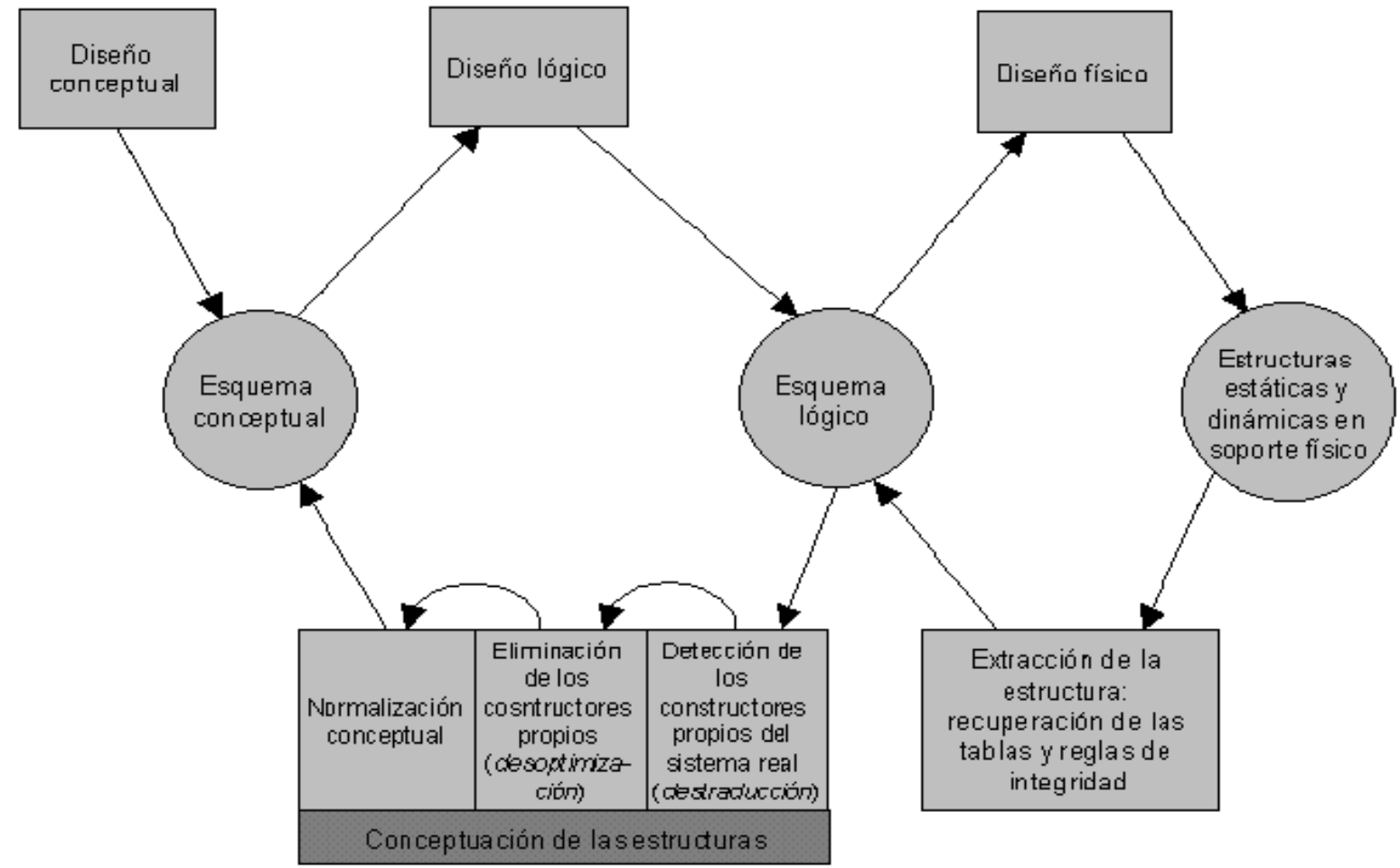
Ingeniería inversa de bases de datos relacionales

Método de Hainaut et al. (II)

- Fase 2) Conceptuación de las estructuras
  - Detectar constructores propios del sistema real
  - Reemplazarlos por constructores independientes
  - Detectar y eliminar constructores no semánticos
  - Normalización conceptual (se obtienen estructuras de alto nivel transformadas en la fase anterior)

Ingeniería inversa de bases de datos relacionales

Método de Hainaut et al. (III)



Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (I)

- Fase 1. Agrupación de tablas.
  - Identificación de claves primarias

Determinar los atributos que forman parte de la PK de cada tabla y resolver los posibles conflictos potenciales de nombre que existan entre ellos.

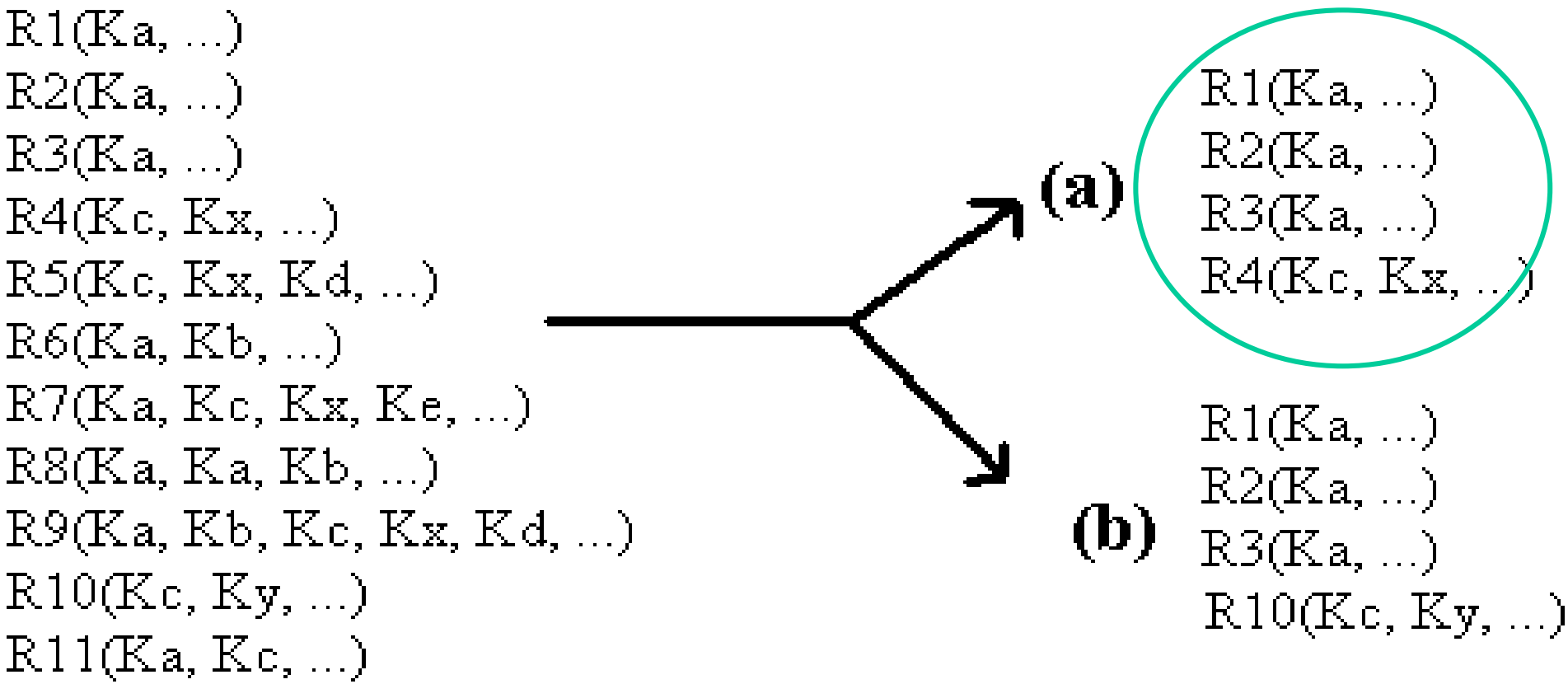
  - Agrupar las tablas en entidades abstractas e interrelaciones

Las tablas se agrupan según los atributos comunes de sus claves primarias, según el siguiente método:

- Seleccionar las tablas cuyas PK: (1) no contienen la PK de otra tabla; (2) son disjuntas o iguales entre sí.
- Colocar las tablas con las mismas PK en el mismo grupo *(entidad abstracta),* de manera que no haya grupos que contengan relaciones con PK disjuntas.
- Añadir cada tabla restante a una entidad abstracta, si al menos un atributo de la PK pertenece a la entidad, y los atributos restantes de la PK no aparecen en ninguna otra entidad.
- Crear un nuevo grupo de tablas cuyos atributos de la PK pertenezcan a las mismas entidades abstractas. Estos grupos se llaman *interrelaciones abstractas*.

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (II)



Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (III)

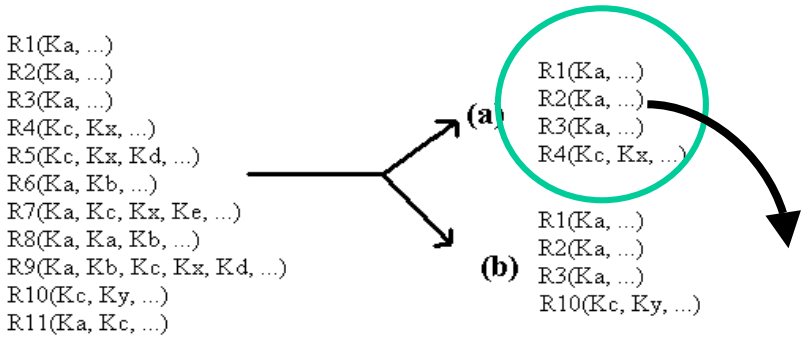
- Fase 1. Agrupación de tablas.
  - Identificación de claves primarias

  Determinar los atributos que forman parte de la PK de cada tabla y resolver los posibles conflictos potenciales de nombre que existan entre ellos.
  - Agrupar las tablas en entidades abstractas e interrelaciones

  Las tablas se agrupan según los atributos comunes de sus claves primarias, según el siguiente método:
    - Seleccionar las tablas cuyas PK: (1) no contienen la PK de otra tabla; (2) son disjuntas o iguales entre sí.
    - Colocar las tablas con las mismas PK en el mismo grupo, de manera que no haya grupos que contengan relaciones con PK disjuntas.
    - Añadir cada tabla restante a una entidad abstracta *(entidad abstracta)*, si al menos un atributo de la PK pertenece a la entidad, y los atributos restantes de la PK no aparecen en ninguna otra entidad.
    - Crear un nuevo grupo de tablas cuyos atributos de la PK pertenezcan a las mismas entidades abstractas. Estos grupos se llaman *interrelaciones abstractas*.

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (IV)



R1(Ka, ...)
R2(Ka, ...) **=AE1**
R3(Ka, ...)

R4(Kc, Kx, ...) **=AE2**

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (V)

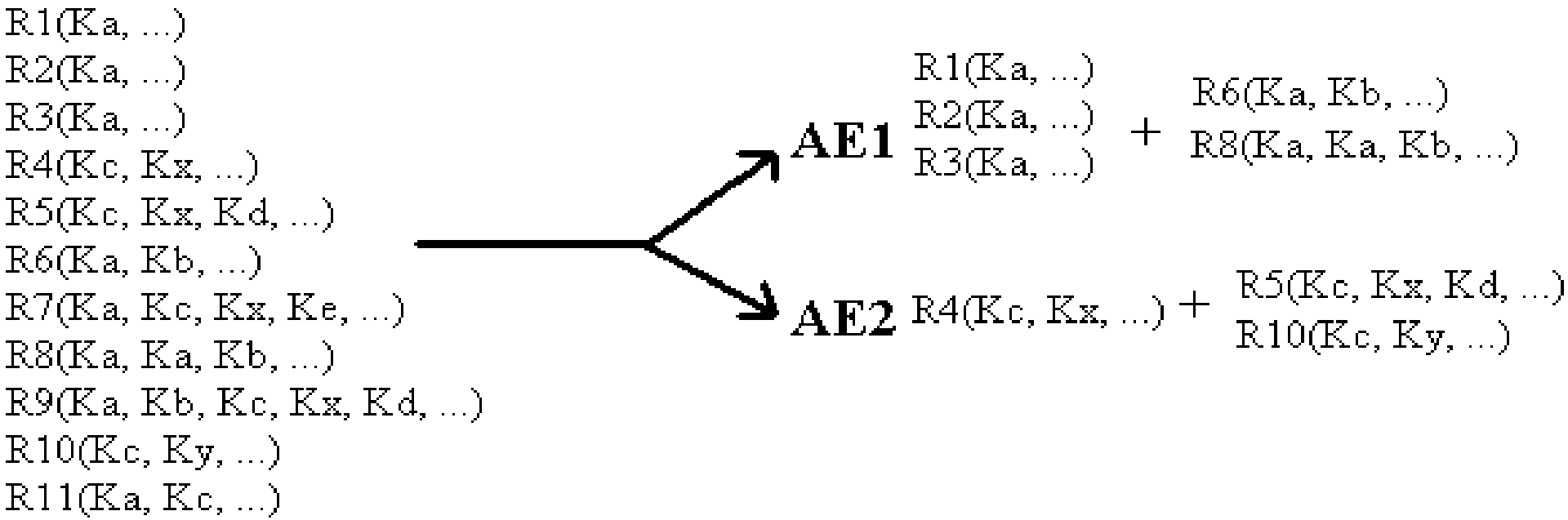
- Fase 1. Agrupación de tablas.
  - Identificación de claves primarias

  Determinar los atributos que forman parte de la PK de cada tabla y resolver los posibles conflictos potenciales de nombre que existan entre ellos.
  - Agrupar las tablas en entidades abstractas e interrelaciones

  Las tablas se agrupan según los atributos comunes de sus claves primarias, según el siguiente método:
    - Seleccionar las tablas cuyas PK: (1) no contienen la PK de otra tabla; (2) son disjuntas o iguales entre sí.
    - Colocar las tablas con las mismas PK en el mismo grupo, de manera que no haya grupos que contengan relaciones con PK disjuntas.
    - Añadir cada tabla restante a una entidad abstracta *(entidad abstracta)*, si al menos un atributo de la PK pertenece a la entidad, y los atributos restantes de la PK no aparecen en ninguna otra entidad.
    - Crear un nuevo grupo de tablas cuyos atributos de la PK pertenezcan a las mismas entidades abstractas. Estos grupos se llaman *interrelaciones abstractas*.

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (VI)



Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (VII)

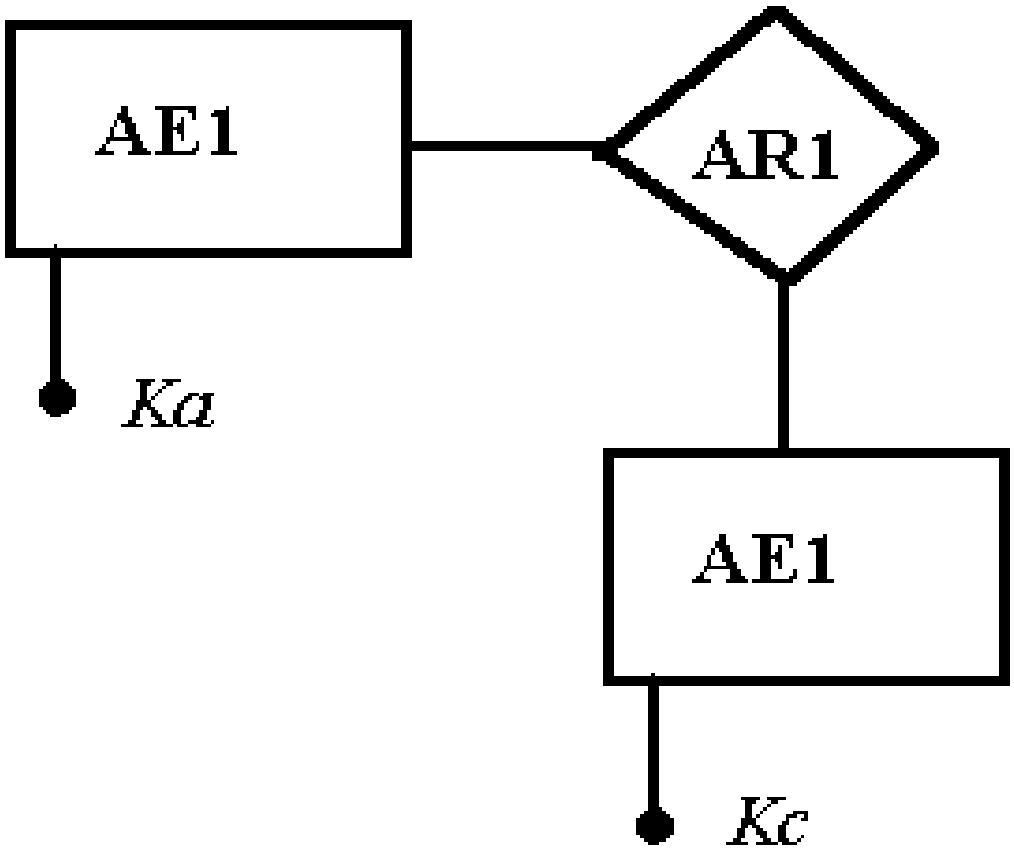
- Fase 1. Agrupación de tablas.
  - Identificación de claves primarias

  Determinar los atributos que forman parte de la PK de cada tabla y resolver los posibles conflictos potenciales de nombre que existan entre ellos.
  - Agrupar las tablas en entidades abstractas e interrelaciones

  Las tablas se agrupan según los atributos comunes de sus claves primarias, según el siguiente método:
    - Seleccionar las tablas cuyas PK: (1) no contienen la PK de otra tabla; (2) son disjuntas o iguales entre sí.
    - Colocar las tablas con las mismas PK en el mismo grupo, de manera que no haya grupos que contengan relaciones con PK disjuntas.
    - Añadir cada tabla restante a una entidad abstracta *(entidad abstracta)*, si al menos un atributo de la PK pertenece a la entidad, y los atributos restantes de la PK no aparecen en ninguna otra entidad.
    - Crear un nuevo grupo de tablas cuyos atributos de la PK pertenezcan a las mismas entidades abstractas. Estos grupos se llaman *interrelaciones abstractas*.

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (VIII)



Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (VII)

- Fase 2. Refinamiento de elementos abstractos.

  La Ing. Inv. de las tablas de una entidad abstracta depende sólo de esa entidad, y a cada entidad abstracta se le puede aplicar un proceso de Ing. Inv. diferente.

Por tanto, para detectar...	*...se puede usar la técnica de:*
Generalizaciones	Petit et al. (1996)
Entidades fuertes	Chiang et al. (1994)
Entidades débiles	Chiang et al. (1994)
Interrelaciones	Christian (1996)
Agregaciones	Chiang et al. (1994)

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (IX)

- Fase 3. Obtención del esquema final.

  Se integran aquí los diferentes esquemas conceptuales intermedios en un esquema final, que se completa con lo que pudiera faltar.

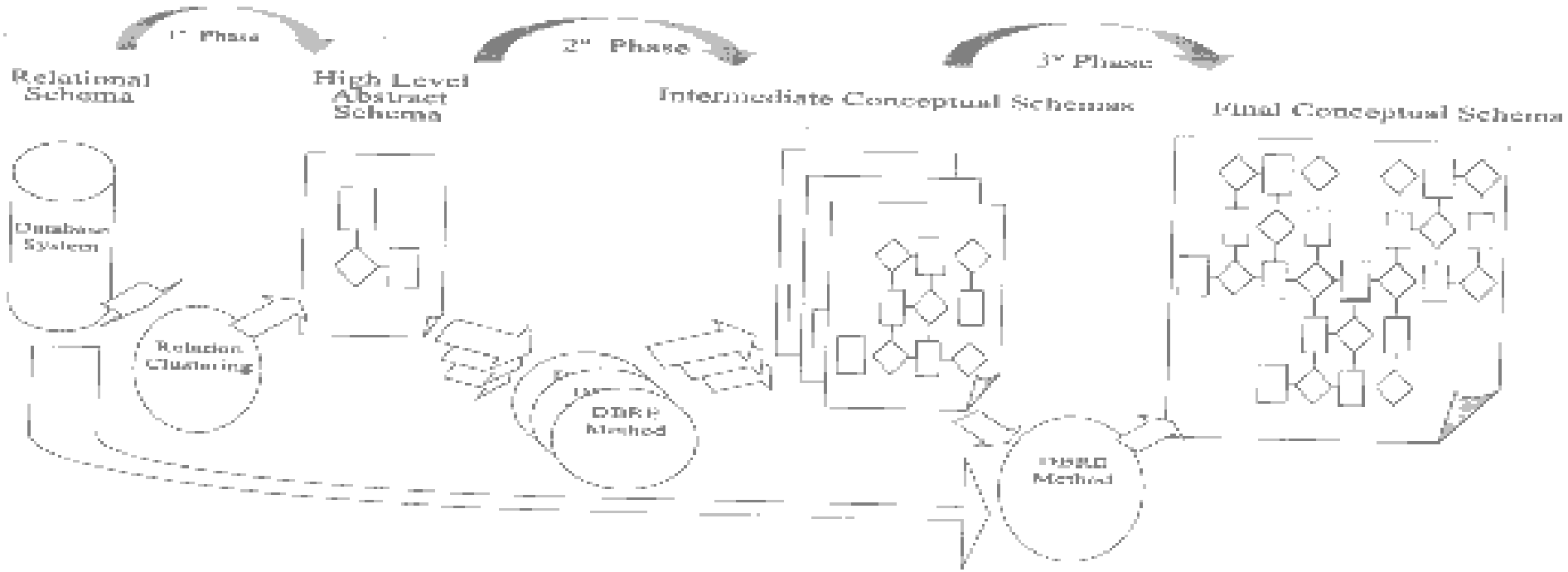


Figure 1. A divide-and-conquer approach for reverse engineering relational databases

Ingeniería inversa de bases de datos relacionales

Método de Sousa et al. (IX)

- Ventajas.
  - Simplifica el proceso de comprensión del dominio de la aplicación (*para qué sirve, qué hace*).
  - Facilita los proyectos de Ing. Inv. de BD, porque se reducen la complejidad y el tamaño del problema.
  - Permite la aplicación de diferentes métodos de Ingeniería Inversa a os diferentes conjuntos de elementos abstractos.
  - Se facilita la gestión del proyecto, porque se pueden dedicar diferentes equipos a los diferentes esquemas, para trabajar en paralelo.

Detección de clones

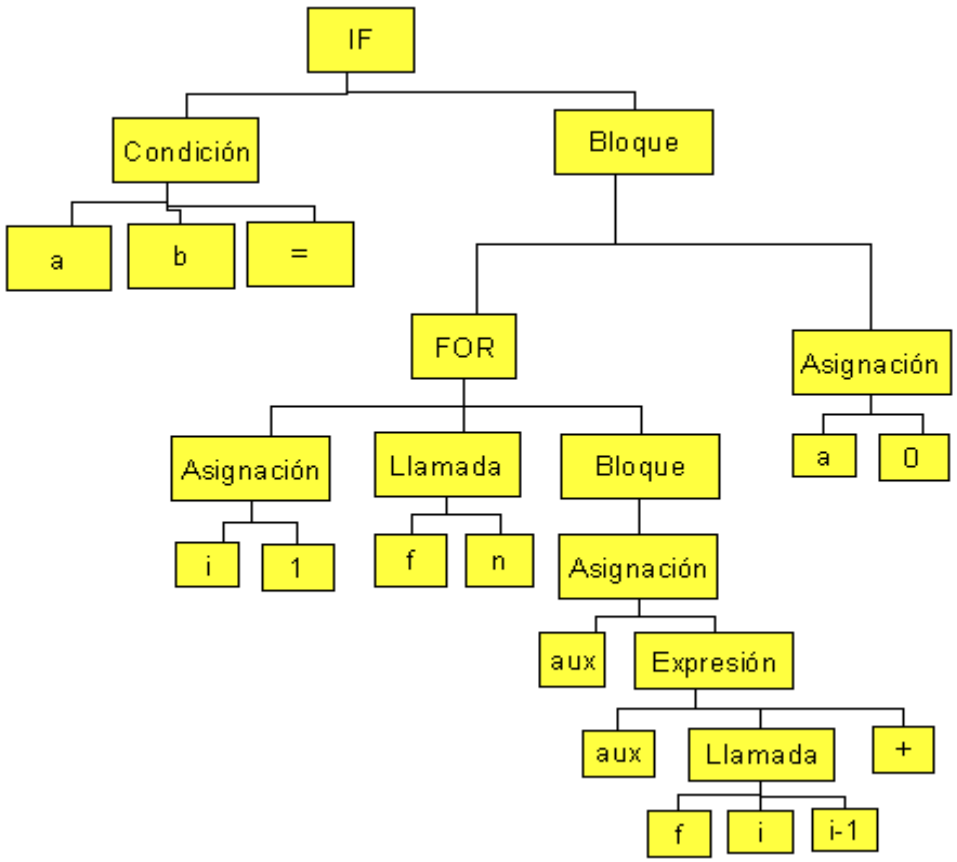
Detección de clones mediante A.S.A. (I) (*Baxter et al., 1998)*

- Causas para la existencia de clones:
  - Copiar y pegar por comodidad
  - Estilos de codificación
  - Operaciones sobre T.A.D.'s
  - Mejora del rendimiento
  - "Accidente"

Detección de clones mediante A.S.A. (II)

- Árboles de Sintaxis Abstracta.

```
...
...
...
IF a=b THEN
    FOR i = 1 TO f(n)
        aux = aux + f(i, i-1)
    NEXT i
    a = 0
END IF
...
...
...
```



Detección de clones mediante A.S.A. (II)

- Consiste en buscar subárboles iguales o parecidos lo más grandes posible:

```
Clones=∅
Para cada s ∈ Subárboles
   Si Hash(s) ≥ PesoMínimo entonces
        TablaHash = TablaHash ∪ {s}
Para cada (s1, s2) que están en la misma entrada de TablaHash
   Si Similitud(s1,s2)≥UmbralDeSimilitud entonces
        Para cada s ∈ Subárboles(s1)
              Si s ∈ Clones entonces
                    Clones=Clones - {s}
        Para cada s ∈ Subárboles(s2)
              Si s ∈ Clones entonces
                    Clones=Clones - {s}
        Clones=Clones ∪ {s1} ∪ {s2}
```