

DISEÑO DE SISTEMAS

- Diagramas de Estructura
- Tablas de interfaz
- Estrategias de diseño
 - Transformación
 - Transacción
- Atributos de calidad de un diseño
- Metodologías de diseño
 - Modelo de Jackson
 - Metodología de Warnier

RELACION ENTRE LAS ACTIVIDADES DE DISEÑO

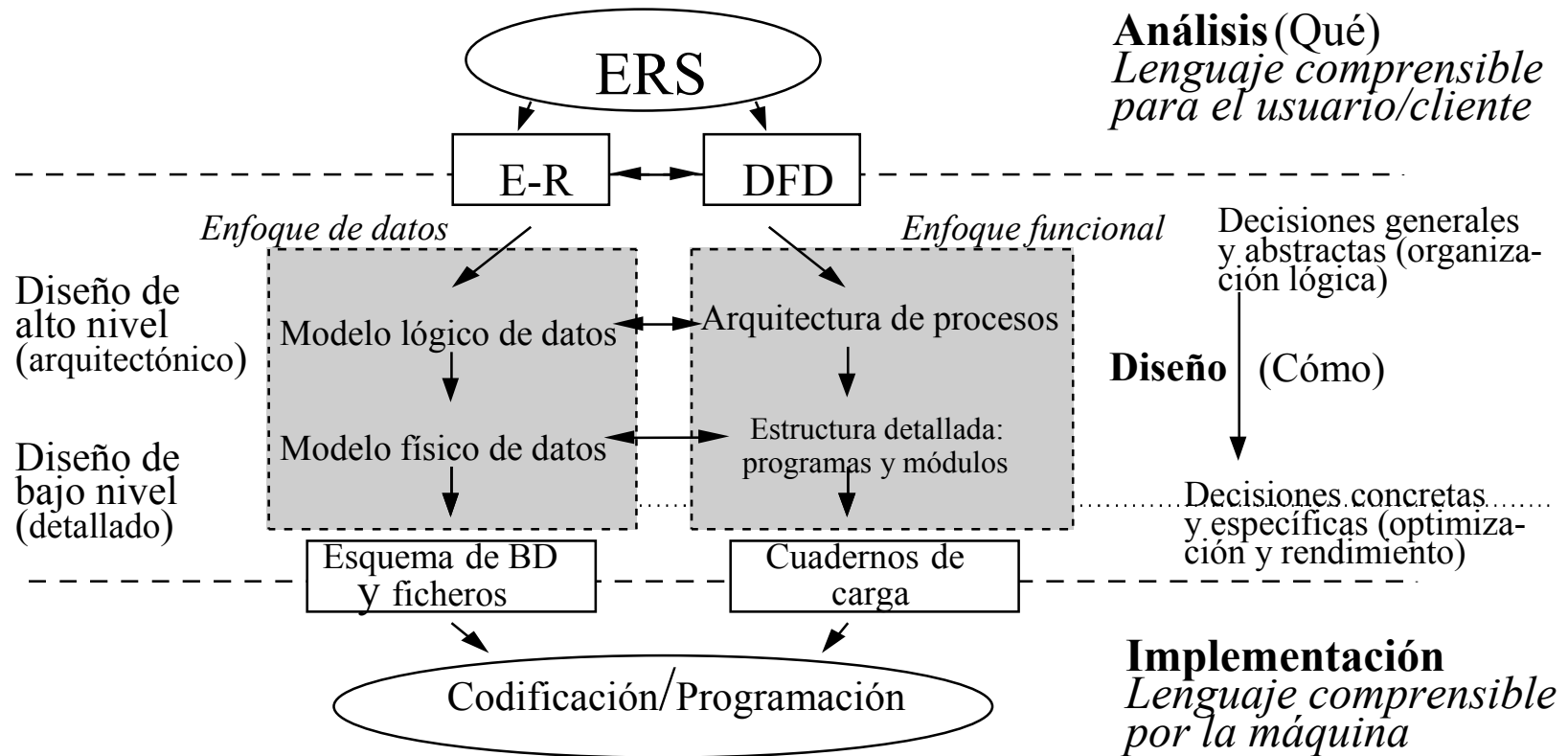


DIAGRAMA DE ESTRUCTURA

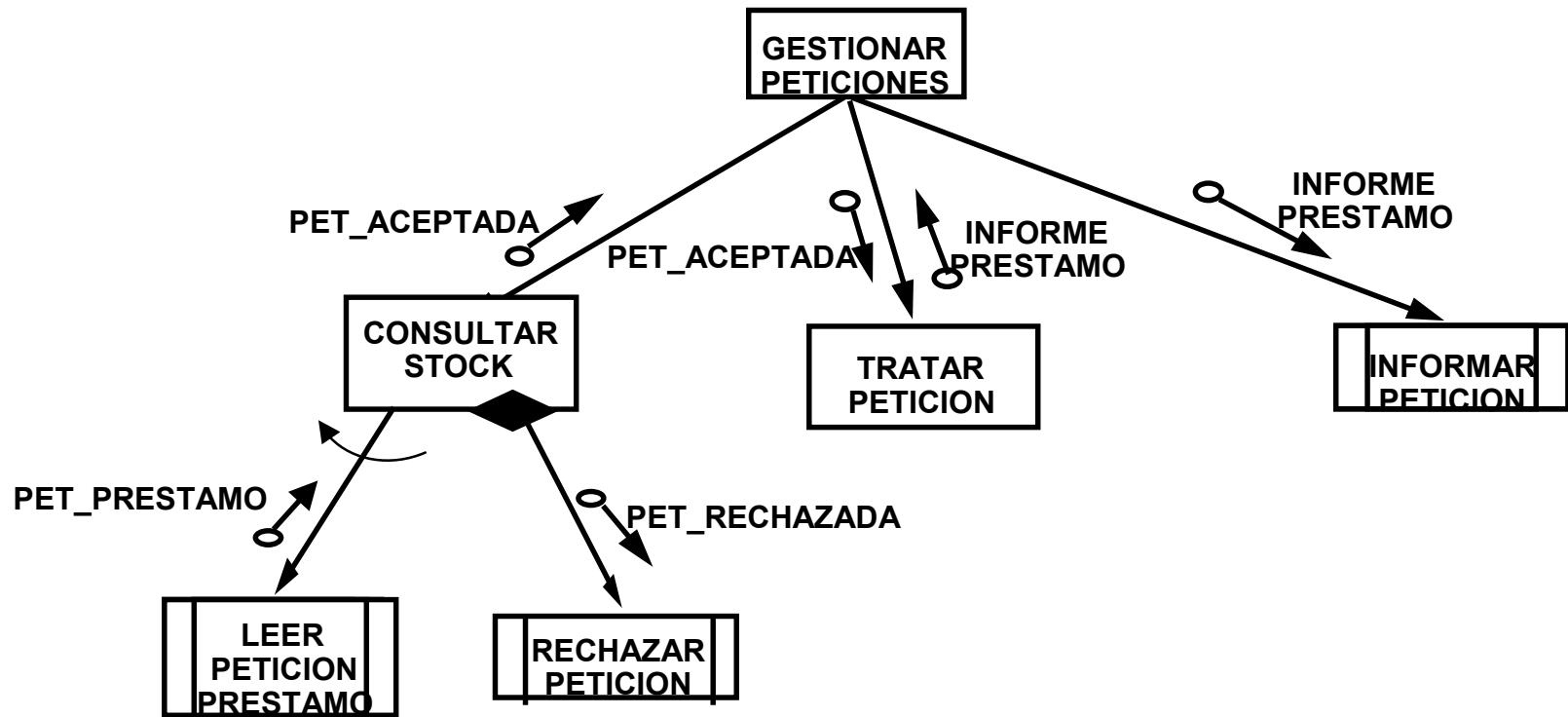


DIAGRAMA DE ESTRUCTURA

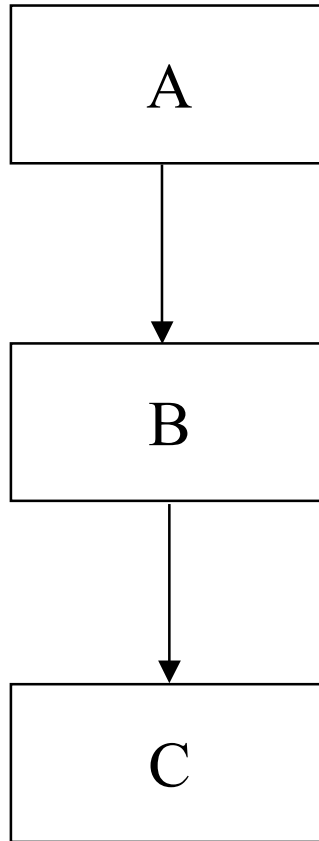


DIAGRAMA DE ESTRUCTURA

Concepto de módulo

- Según la Asociación Española para el Control de Calidad [AECC, 1986], un módulo es la *parte lógica separable de un programa*
- Según Yourdon [YOURDON y CONSTANTINE, 1979], un módulo *es una secuencia contigua de sentencias de programa, limitada por delimitadores y que tiene un identificador global*
 - Según Fenton [FENTON, 1991], un módulo *puede ser cualquier objeto que, en un nivel de abstracción dado, queramos considerar como un concepto simple*
- En la teoría del diseño estructurado [PAGE-JONES, 1988], un módulo *es aquella parte de código que se puede llamar*

DIAGRAMA DE ESTRUCTURA

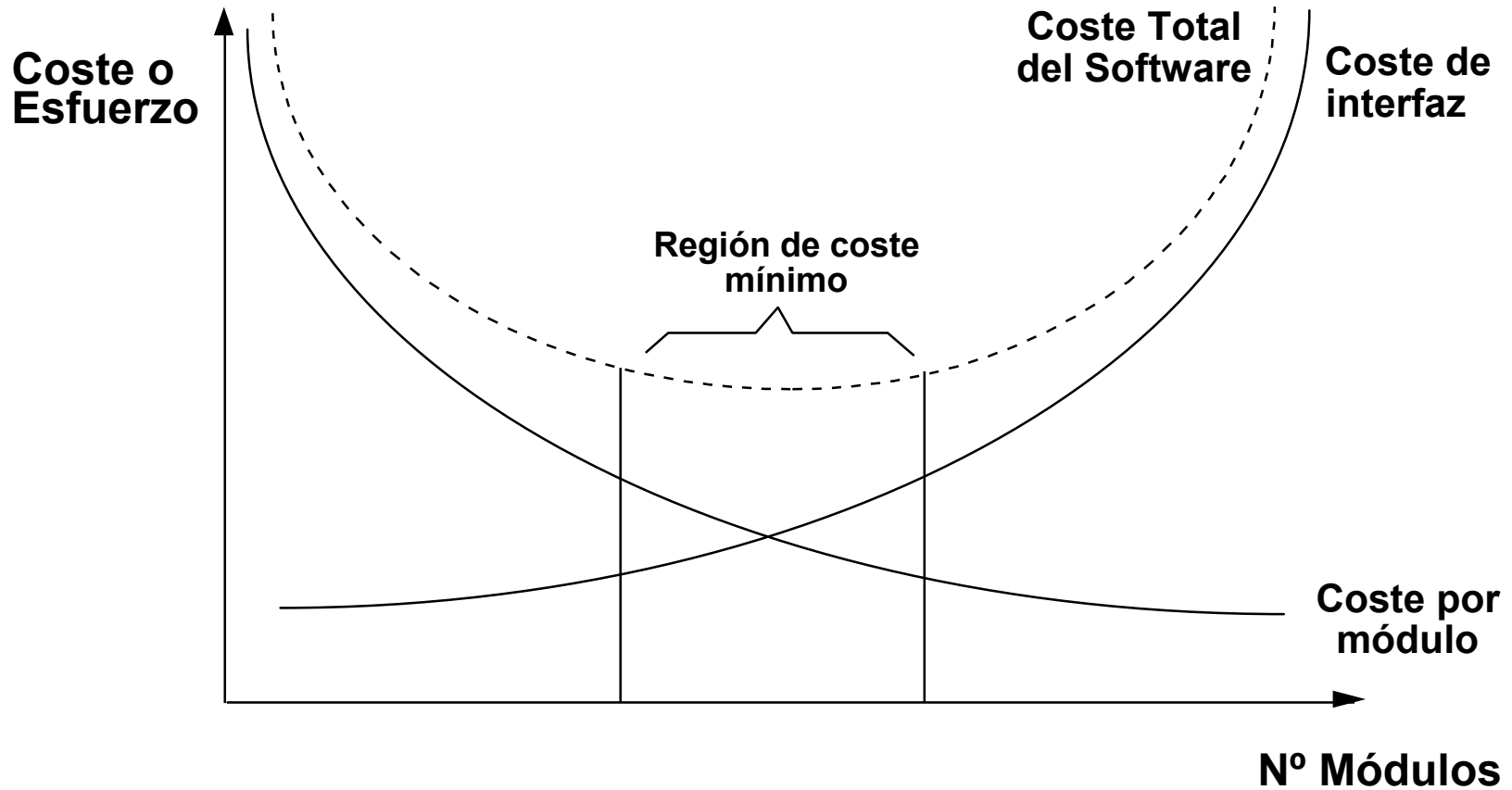


DIAGRAMA DE ESTRUCTURA

CONEXION ENTRE MODULOS

Un sistema está compuesto por módulos organizados jerárquicamente, cooperando y comunicándose entre sí para realizar una tarea. La llamada de un módulo se representa con una flecha

DIAGRAMA DE ESTRUCTURA

COMUNICACION ENTRE MODULOS

La comunicación intermodular se realiza a través de los datos y los *flags*. Los datos se procesan; por el contrario, los *flags* sólo sirven como valores de condición para comunicar condiciones entre los módulos. Otra diferencia es que los datos están relacionados con el problema y son importantes para el mundo exterior, mientras que los *flags* sólo importan para la comunicación de información.

TABLA DE INTERFAZ

- 1.- El módulo llamado
- 2.- Cada parámetro formal
- 3.- Si el parámetro es de entrada
(marcando la columna correspondiente)
- 4.- Si el parámetro es de salida
(marcando la columna correspondiente)
- 5.- El uso de cada parámetro
- 6.- El significado de cada parámetro

TABLA DE INTERFAZ

Módulo	Parámetro Formal	Entrada	Salida	Uso	Significado Parámetro
F(x,y)	x	sí	no	P	Fecha-Nacimiento
	y	no	sí	M	Edad

TABLA DE INTERFAZ

Nemotécnico	Significa
P	El parámetro es PROCESADO: $a = b + 2$
M	El parámetro es MODIFICADO: $a = 3 + b$
T	El parámetro es TRANSFERIDO por el módulo llamado a otro módulo que éste llama, sin modificar su valor
C	El parámetro es usado como una VARIABLE DE CONTROL, quizás para actuar como índice conmutador, como un valor de un flag o para la especificación de una función que es usada por el módulo llamado.
I	El parámetro es TRANSFERIDO a otro módulo, y es MODIFICADO en este segundo módulo

TABLA DE INTERFAZ

Módulo	Parámetro Formal	Entrada	Salida	Uso	Significado Parámetro
TRATAR PETICIÓN	Pet_Aceptada	sí	no	P	Petición Aceptada
	Informe Préstamo	no	sí	I	Informe de Préstamo
INFORMAR PETICIÓN	Informe Préstamo	sí	no	P	Informe de Préstamo

ESTRATEGIAS DE DISEÑO

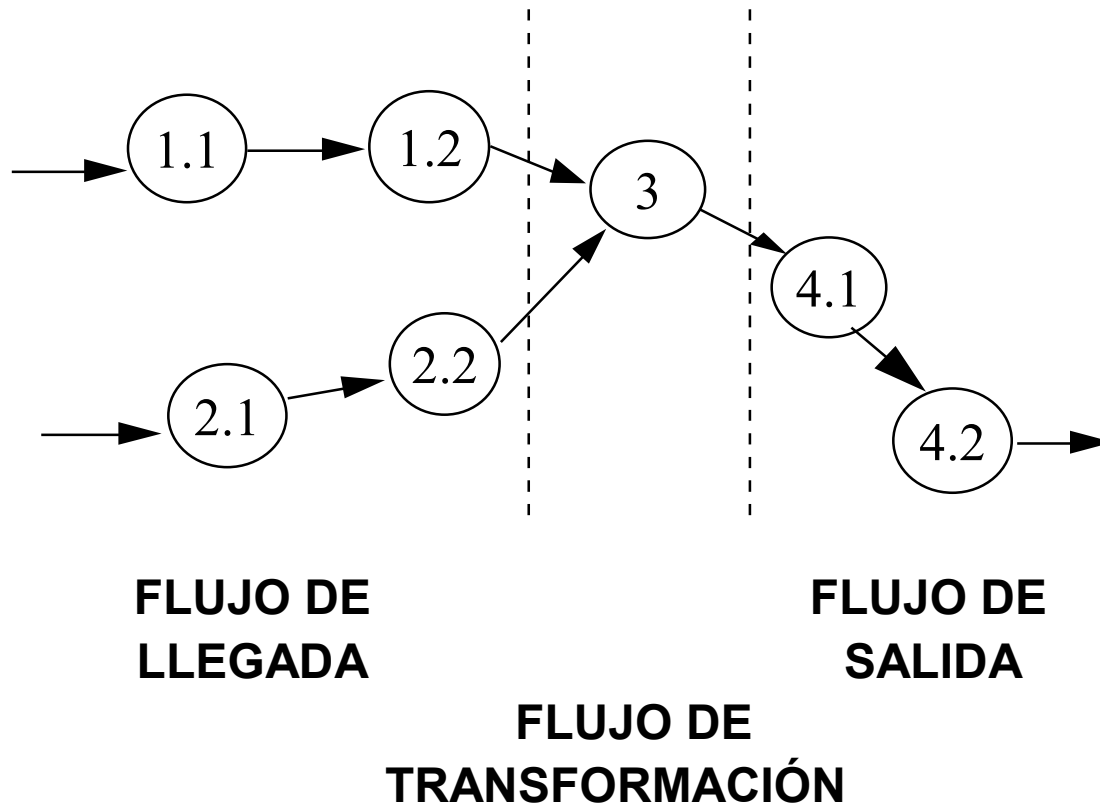
El diseño estructurado ofrece dos estrategias para conseguir una creación rápida de un buen diseño a partir de una ERS:

- **Diseño por Transformación:** los datos entran en el sistema mediante caminos que se denominan flujos de entrada. En el núcleo se produce una transformación de los datos, y finalmente, los datos se mueven por caminos que conducen a la salida.
- **Diseño por transacción:** Existe un centro de transacción que es el centro de flujo, desde el que emanan muchos caminos alternativos de forma exclusiva.

El diseño estructurado permite una transición del DFD a una descripción de diseño de la estructura del programa. Se definen unos pasos que están en función del tipo de flujo de información de que se trate.

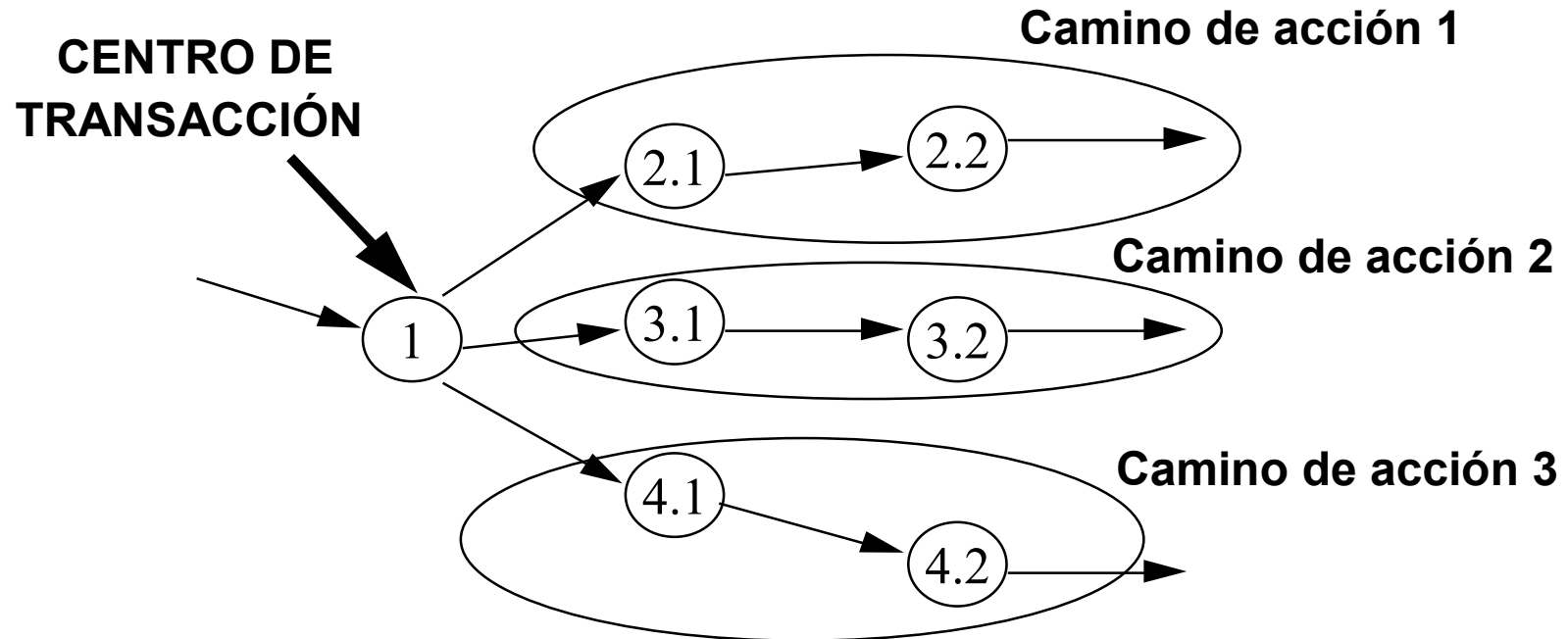
ESTRATEGIAS DE DISEÑO

FLUJO DE TRANSFORMACION



ESTRATEGIAS DE DISEÑO

FLUJO DE TRANSACCION



ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION

1. Revisión del modelo fundamental del sistema

Debe haberse aplicado análisis estructurado (DFD).

Hay que considerar al el DFD expandido (3er nivel).

2. Determinar si el DFD tiene características de transformación o de transacción

La mayoría de flujos se representan como transformaciones.

Si existe un proceso con salidas exclusivas entonces se trata de un problema de transacción..

3. Aislar el centro de transformación, especificando los límites del flujo de llegada y de salida

El centro de transformación es la parte del DFD que contiene las funciones esenciales del sistema.

Los límites están abiertos a interpretación (diseñador) →

diferentes soluciones de diseño según la localización de los límites del flujo.

ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION

4. Realizar el primer corte del diagrama de estructura

Primer nivel de factorización del DE:

módulo principal coordinador, controlador de entrada, controlador del centro de transformación, y módulo controlador de salida de datos del sistema.

Los módulos deben tener nombres significativos.

El nombre del Cm coincide con el nombre del diagrama de contexto

5. Ejecución del segundo nivel de factorización

Se empieza en los límites y se dirige hacia fuera.

Las transformaciones se convierten en módulos.

Se introducen módulos predefinidos que proporcionen las diferentes E/S que necesita/genera el sistema.

6. Refinar la estructura del sistema utilizando medidas y guías de diseño

Se pueden aumentar disminuir el nº de módulos para producir una factorización lógica, con buena calidad, fácil de implementar/probar/mantener.

Refinamientos: están dictados por consideraciones prácticas, sentido común y requisitos del software.

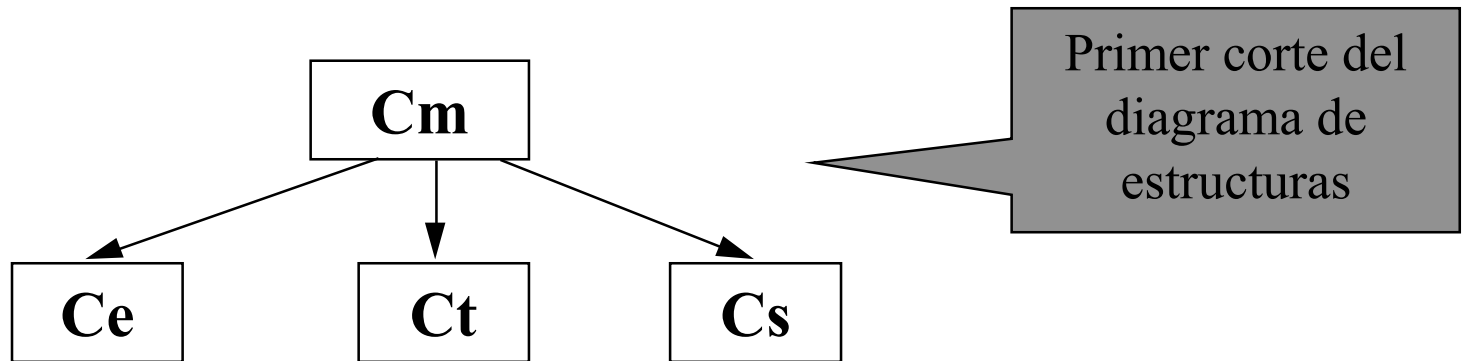
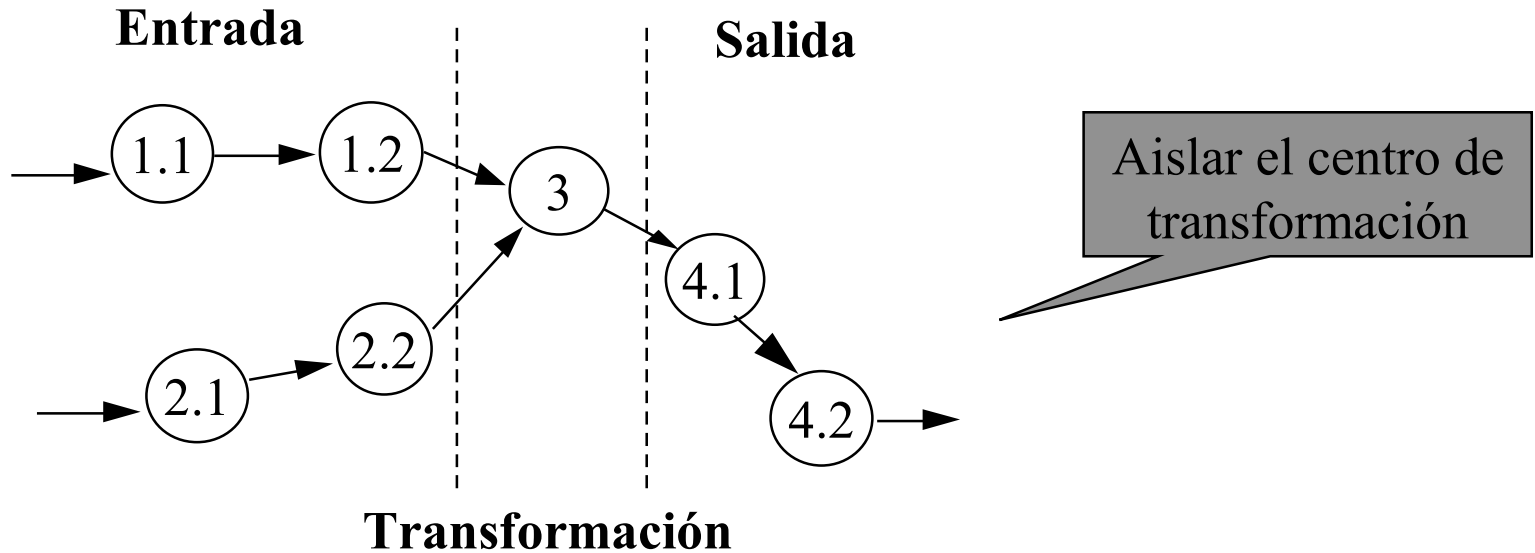
Reflejar los parámetros: datos=flujos de información del DFD; flags=se obtienen de las descripciones de procesos.

7. Asegurarse del trabajo realizado por el diseño obtenido

Se puede revisar el DE comprobando que el orden de ejecución de los módulos es el correcto

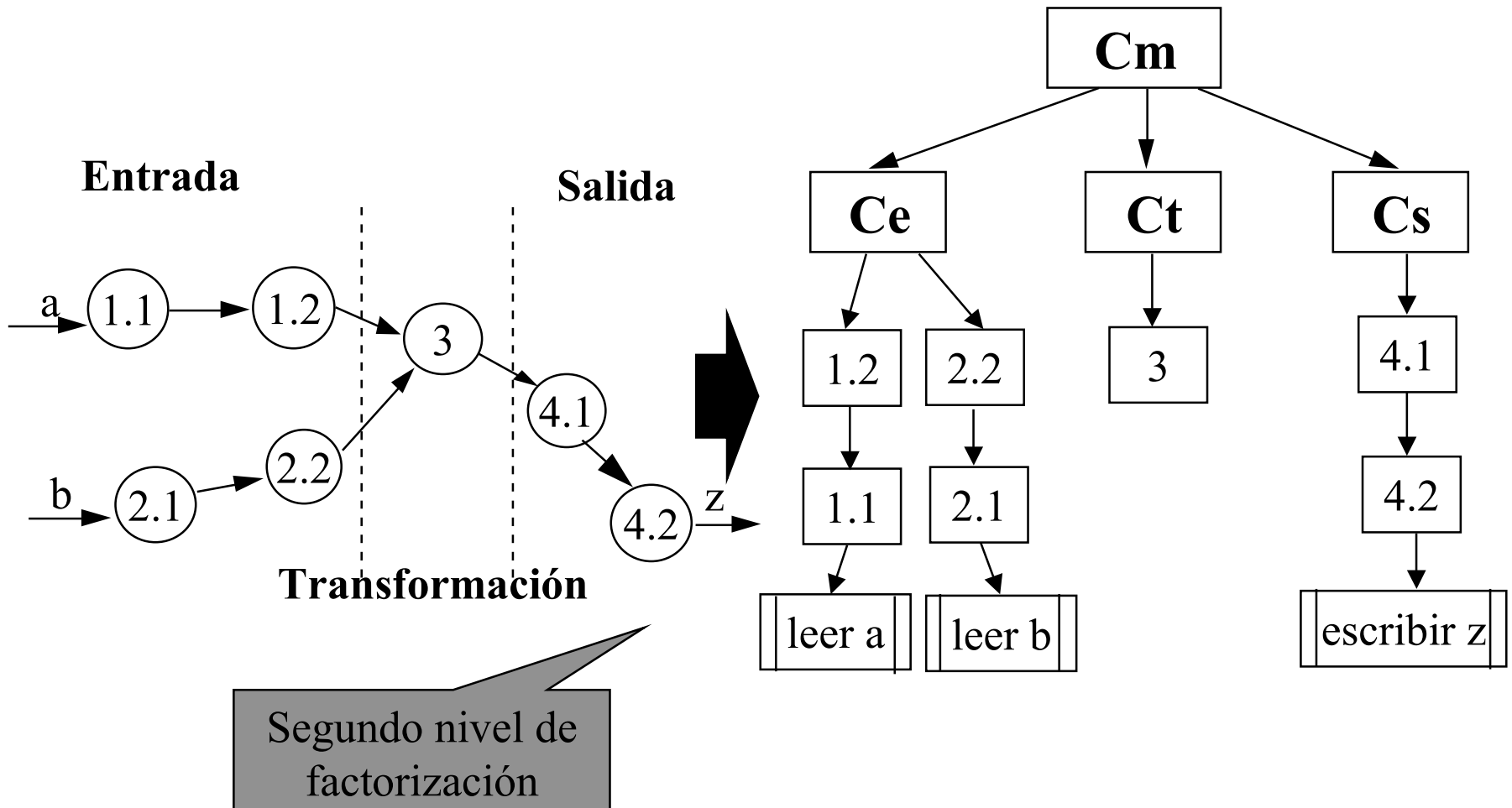
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION



ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION

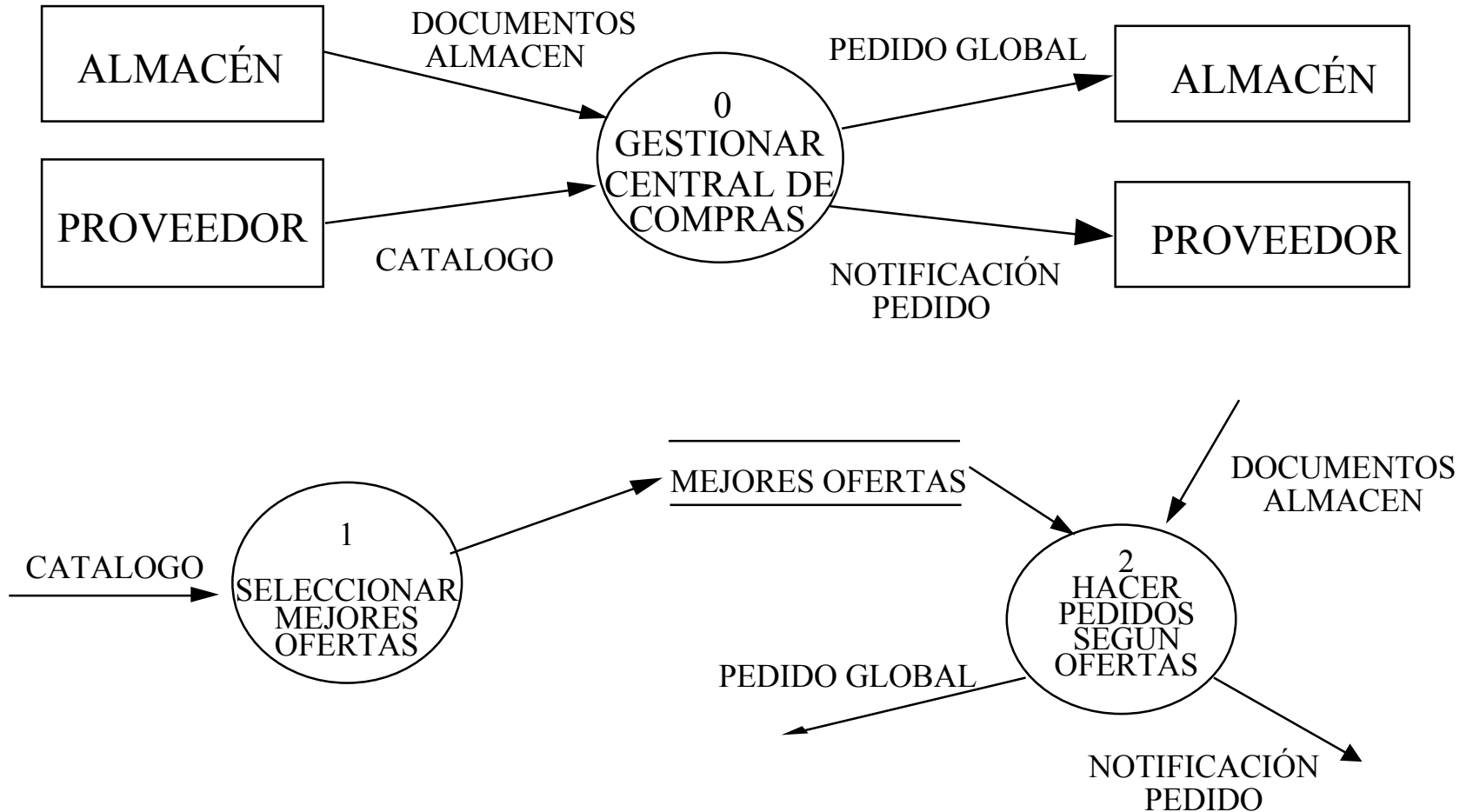


Segundo nivel de factorización

Ejemplo

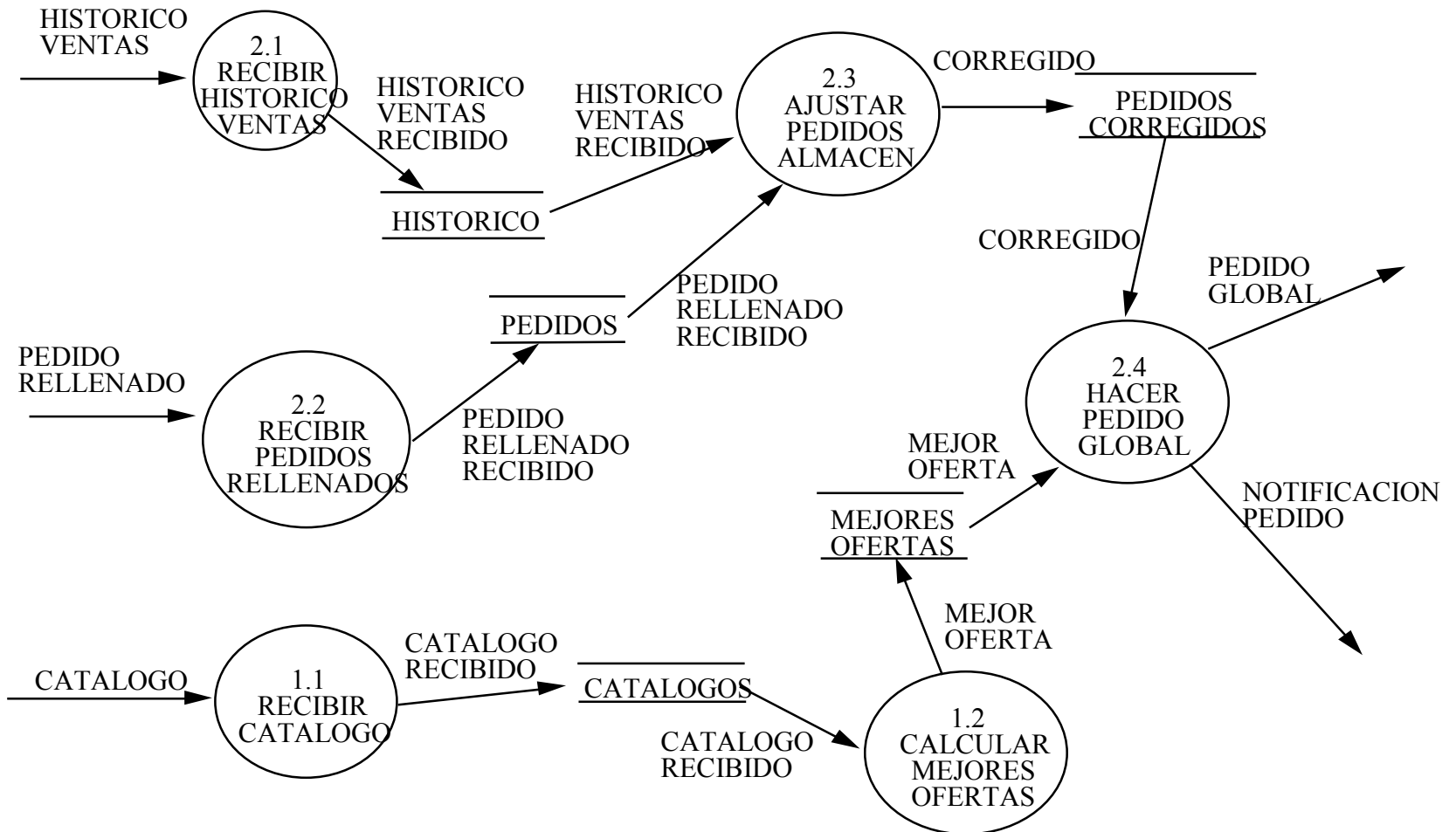
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION



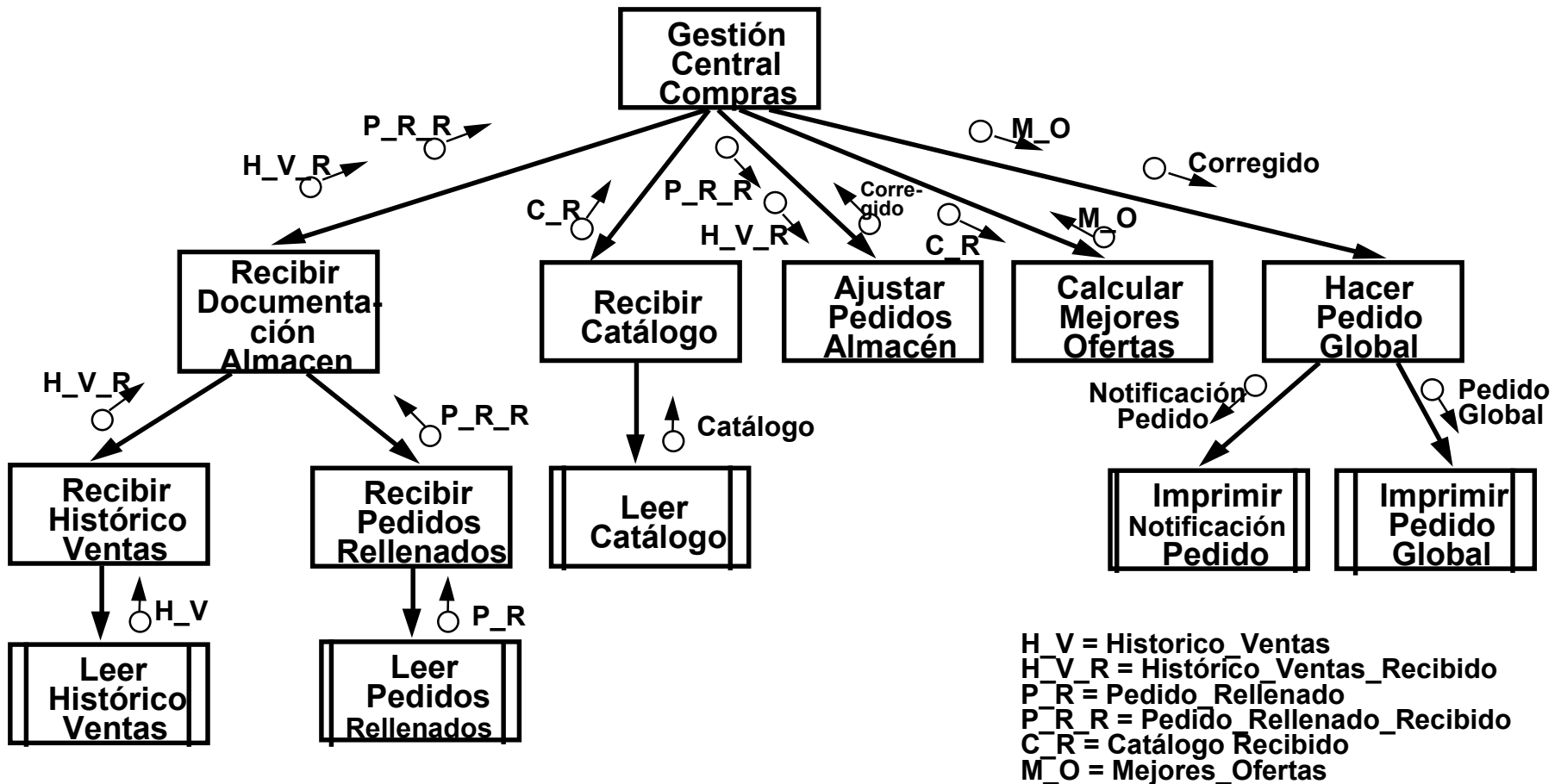
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION



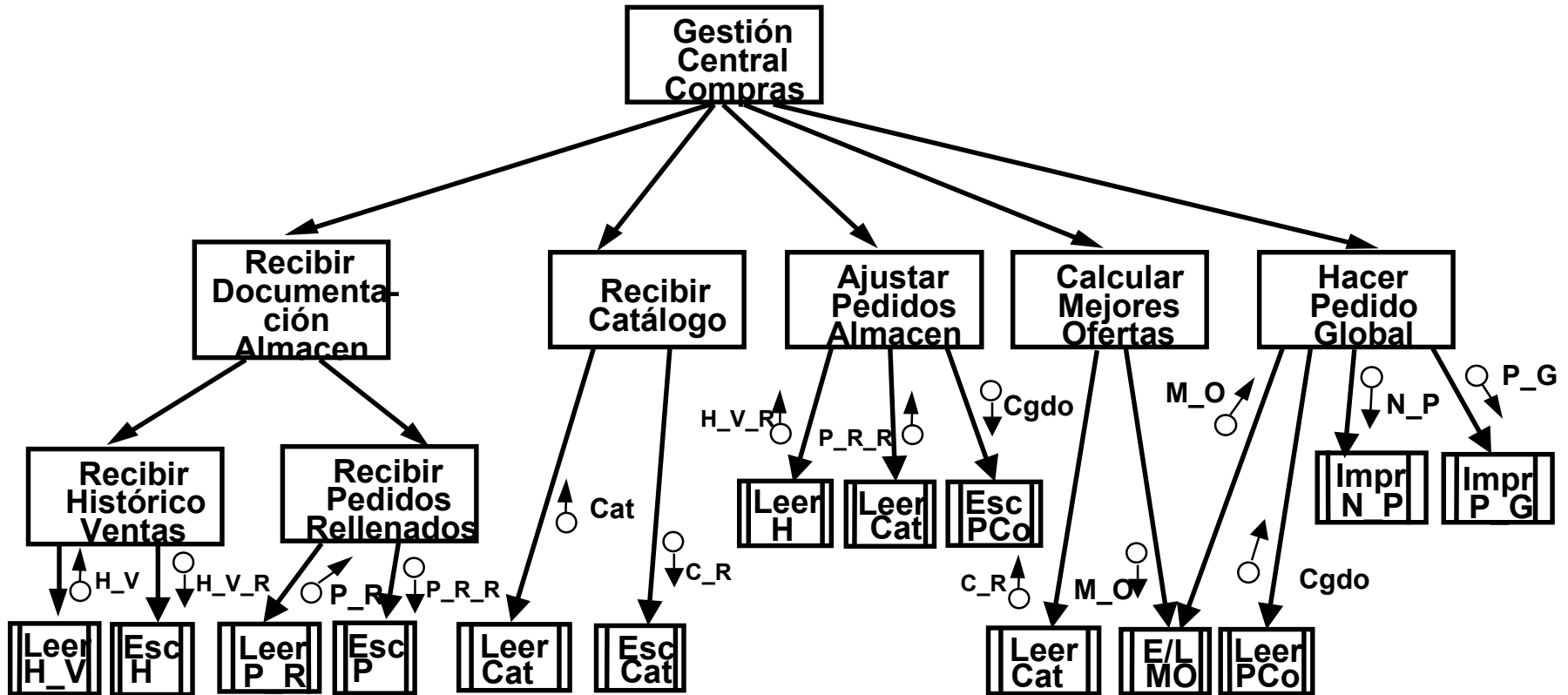
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION



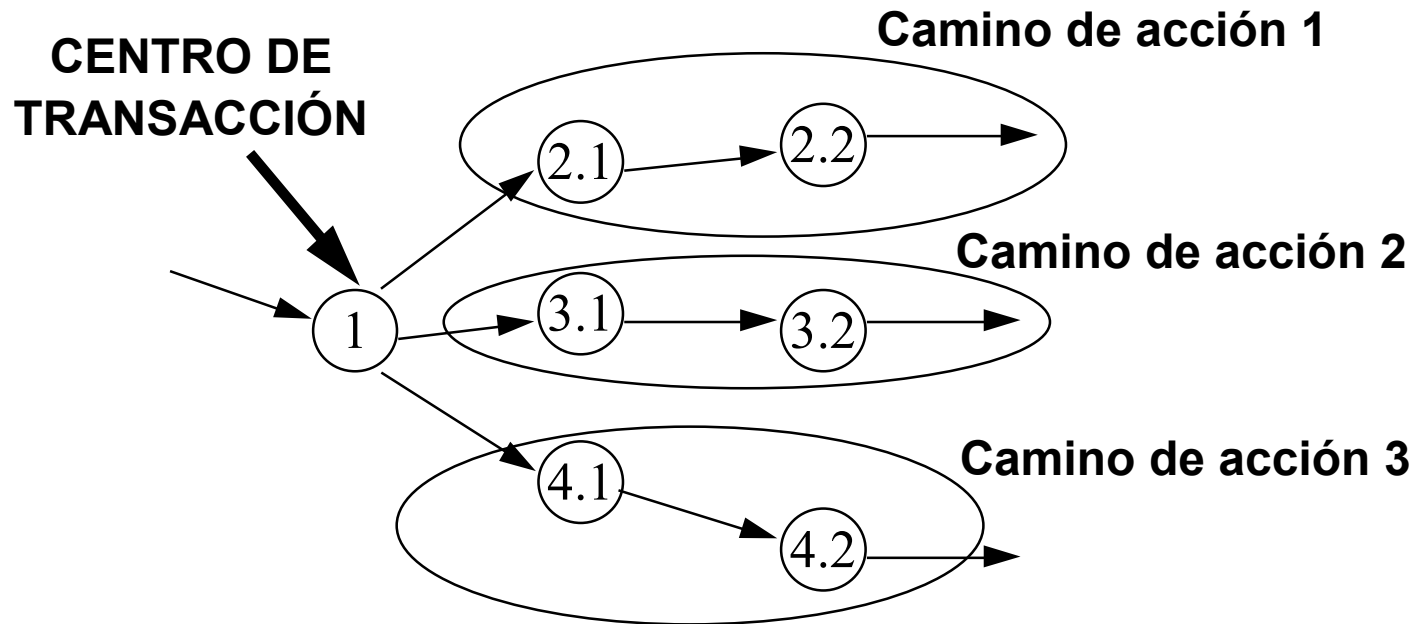
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSFORMACION



ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION

- ✓ Revisión del modelo fundamental del sistema
- ✓ Determinar si el DFD tiene características de transformación o de transacción
- ✓ Identificar el centro de transacción y las características del flujo de cada camino de acción

La posición del centro de transacción puede descubrirse inmediatamente a partir del DFD. El centro de transacción está ligado al origen de varios caminos de información que fluyen de él. La exclusividad no se suele reflejar en el DFD, por lo que hay que conocer los requisitos.

Se identifica el camino de llegada, el centro de transacción y los caminos de acción.

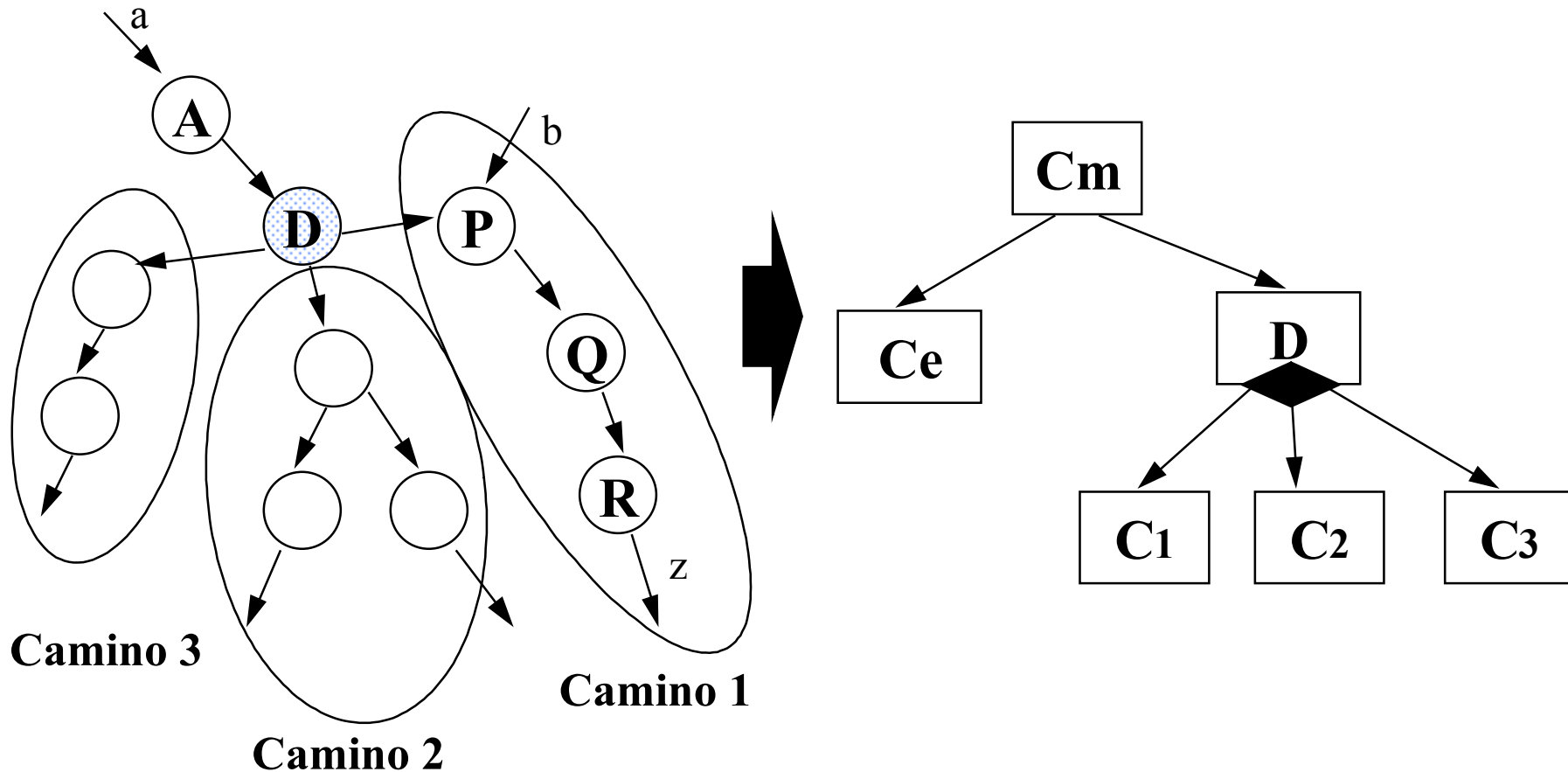
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION

- ✓ Realizar el primer corte del diagrama de estructuras
- ✓ Realizar el segundo nivel de factorización
- ✓ Refinar la estructura del programa
- ✓ Asegurarse del trabajo realizado por el diseño obtenido

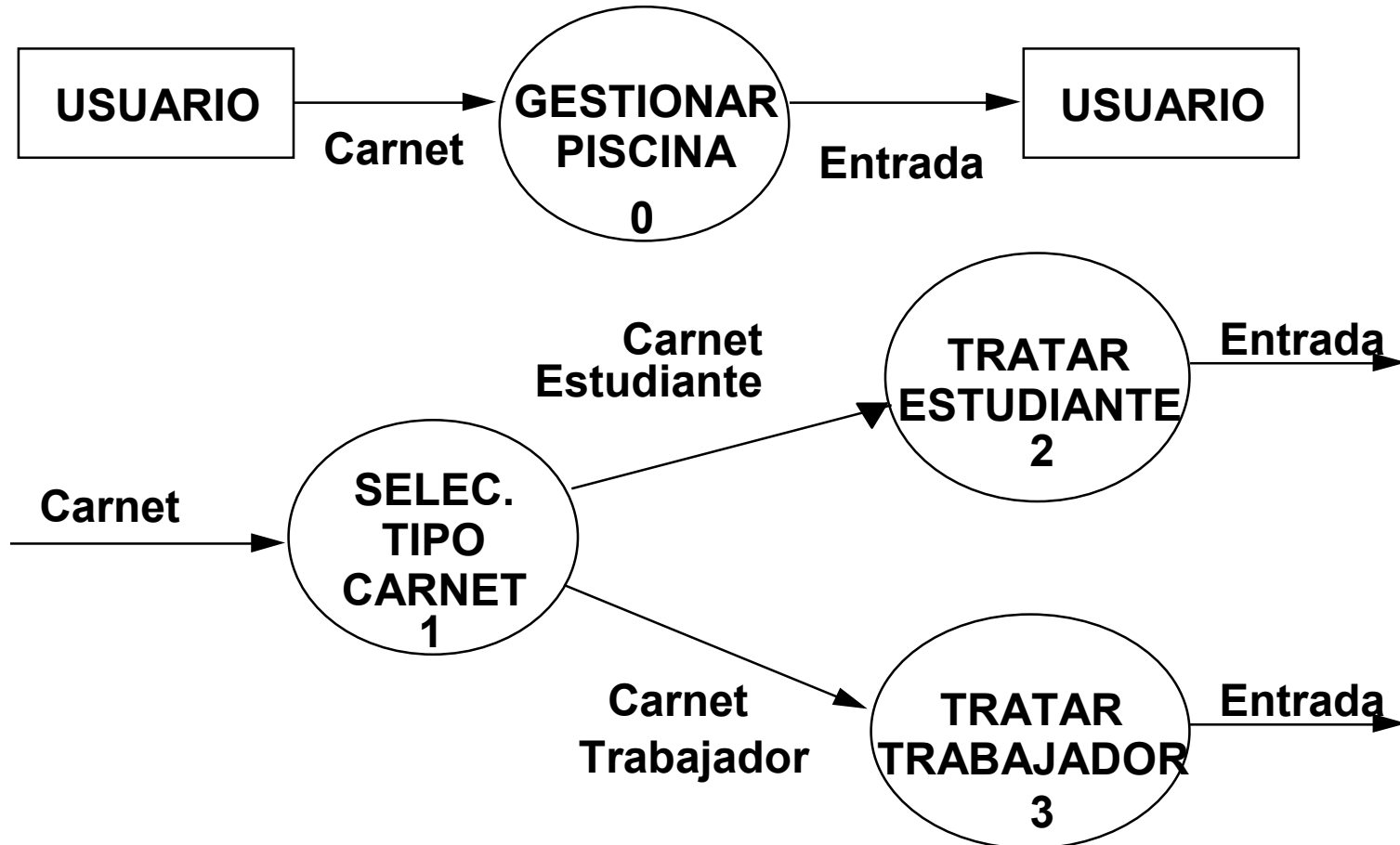
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



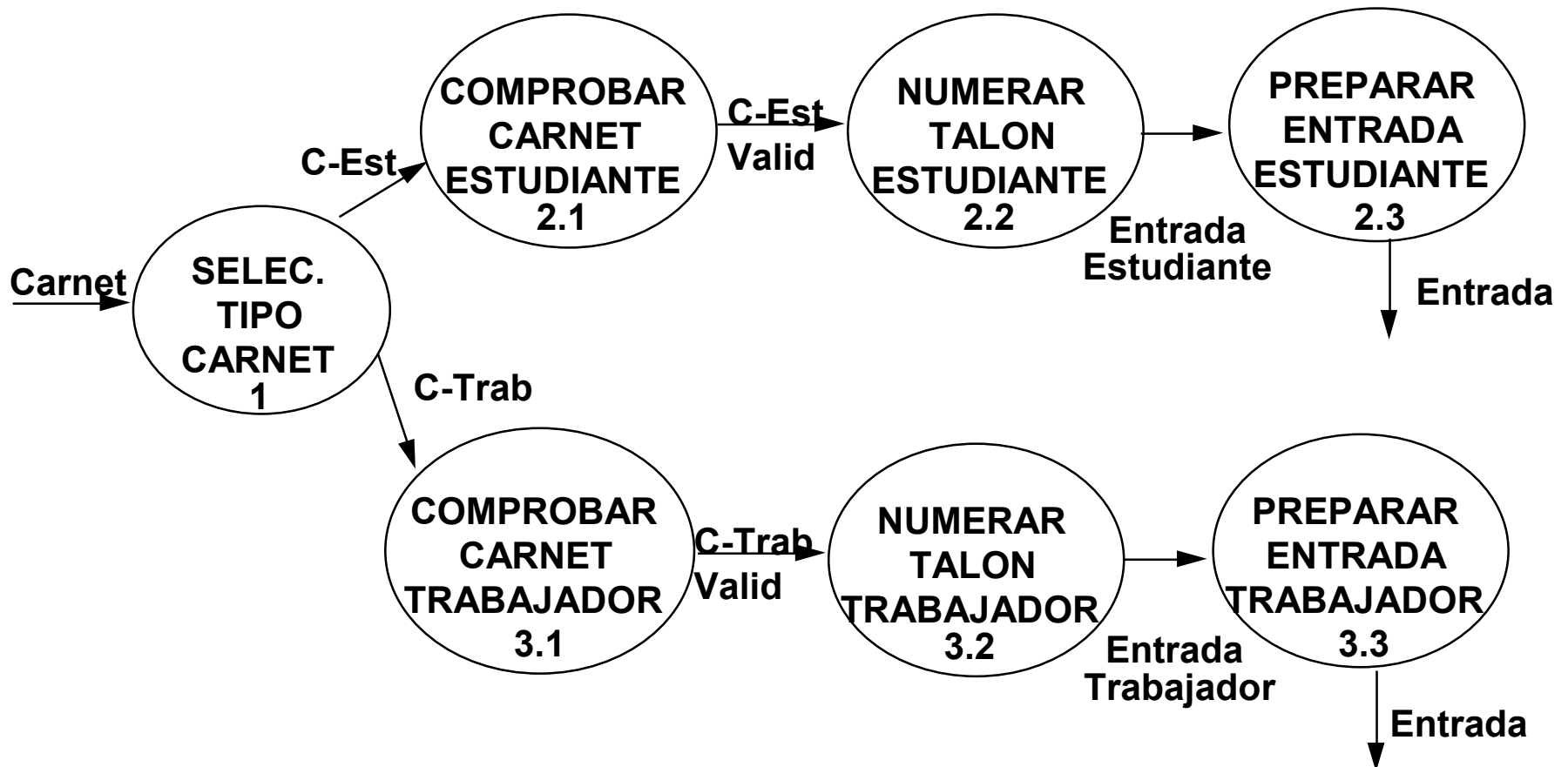
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



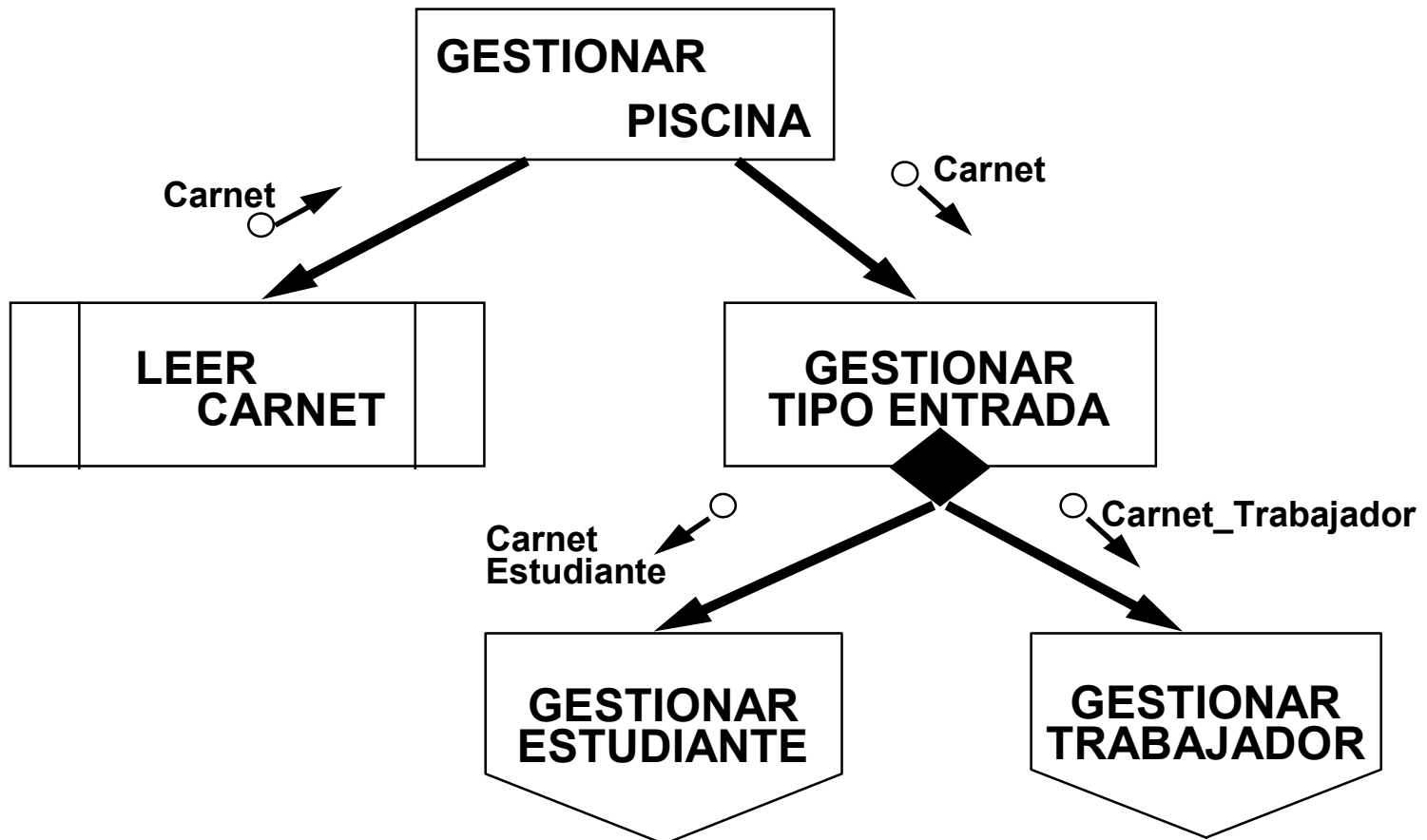
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



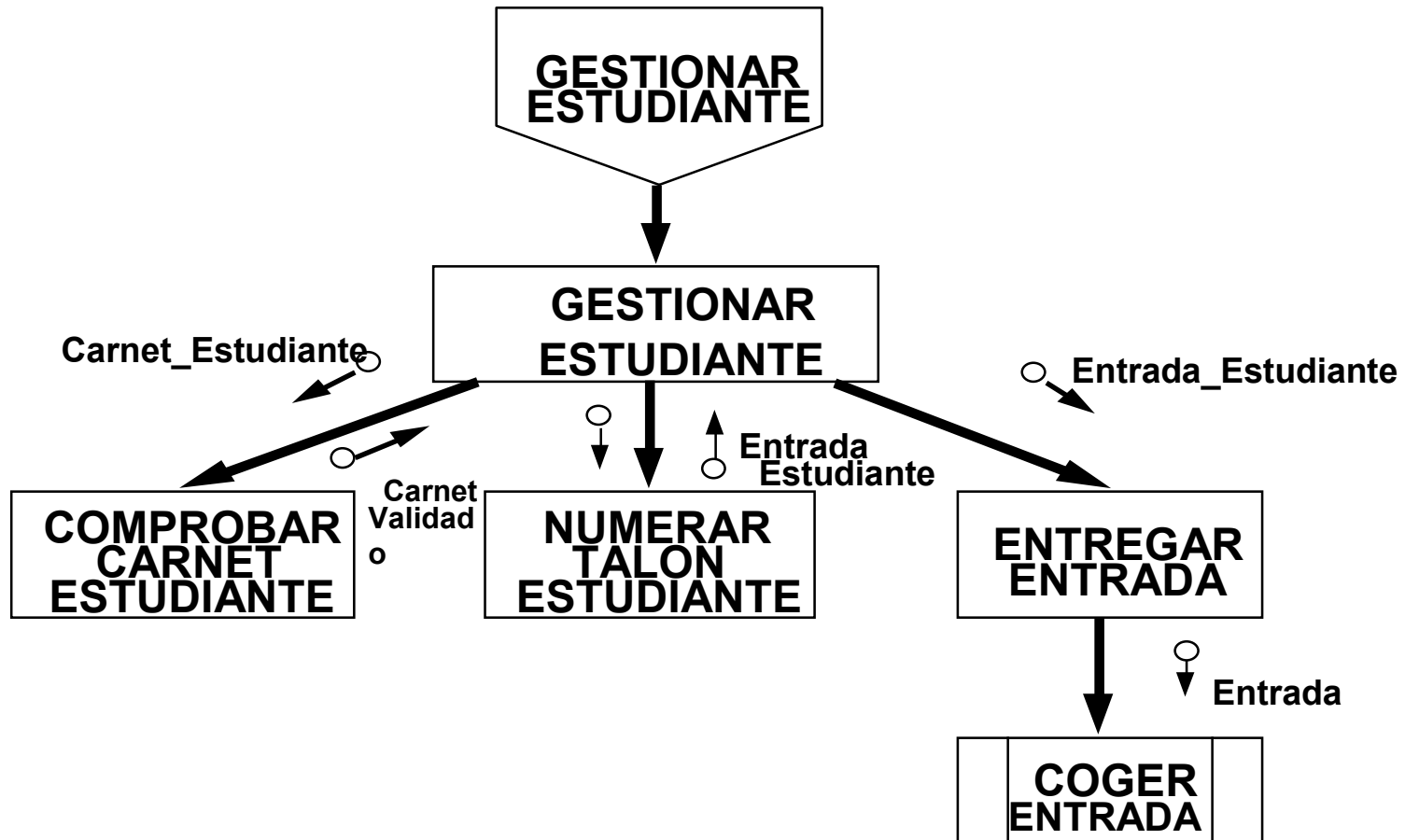
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



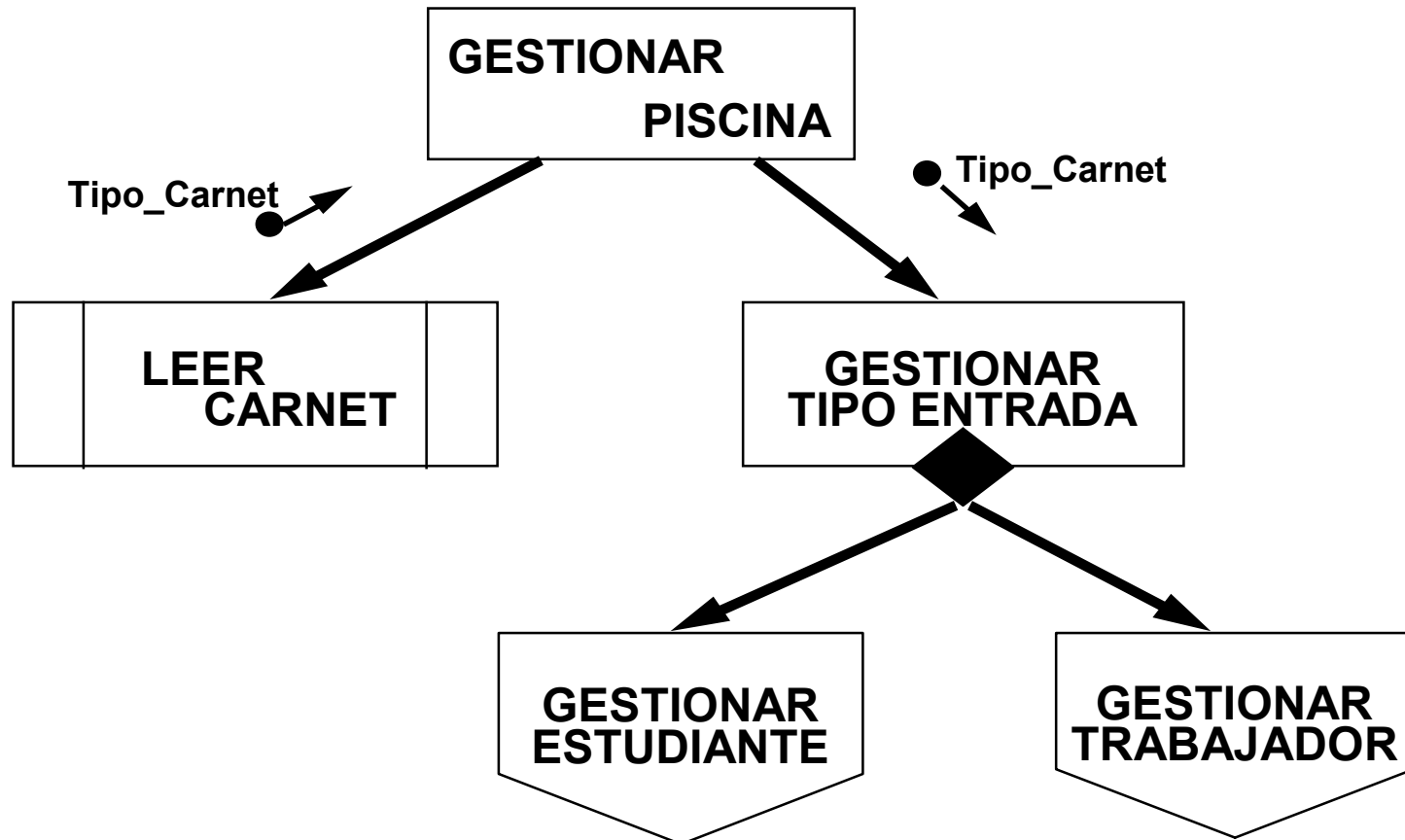
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



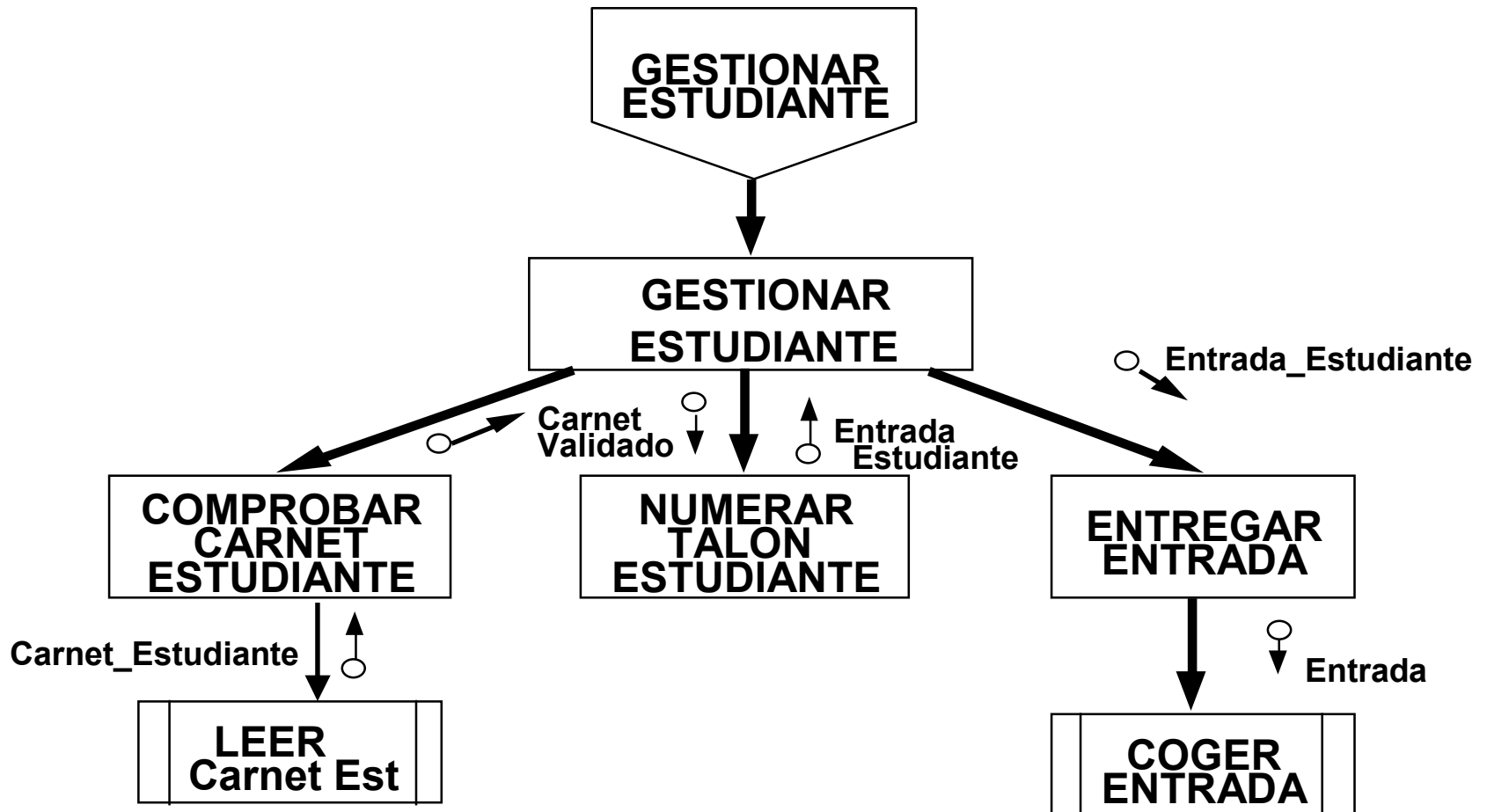
ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



ESTRATEGIAS DE DISEÑO

ANALISIS DE TRANSACCION



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

En cada proyecto se deben decidir cuáles son los requisitos de calidad a cumplir, y decidir los más importantes.

Para asegurar y evaluar la calidad del software, ésta se debe poder medir. Para ello se emplean las MÉTRICAS del software

Aquí nos centramos en las métricas que miden la calidad estructural:

- Cohesión
- Acoplamiento

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

Acoplamiento

Es el grado de interdependencia entre los módulos.

Un buen diseño se caracteriza por un acoplamiento mínimo, es decir, unos módulos tan independientes los unos de los otros como sea posible.

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ESCALA DE ACOPLAMIENTO

NORMAL

MEJOR

- de datos

- por estampado

- de control

EXTERNO

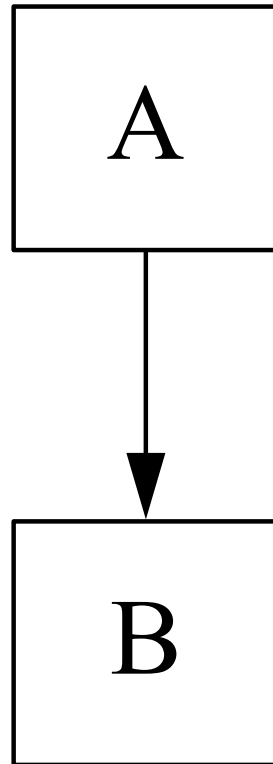
COMÚN

POR CONTENIDO

PEOR

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

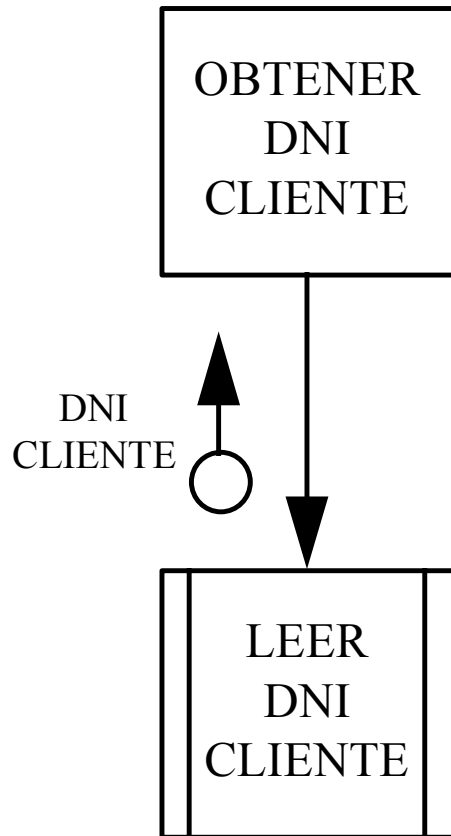
ACOPLAMIENTO NORMAL



Un módulo llama a otro, y no se pasan ningún tipo de información

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO DE DATOS



Los módulos se comunican mediante paso de parámetros. Hay que reducir tanto como sea posible la información que se intercambia entre módulos

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO POR ESTAMPADO



Se necesita el DNI y se pasan todos los datos del Cliente

CLIENTE

OBTENER
DNI
CLIENTE



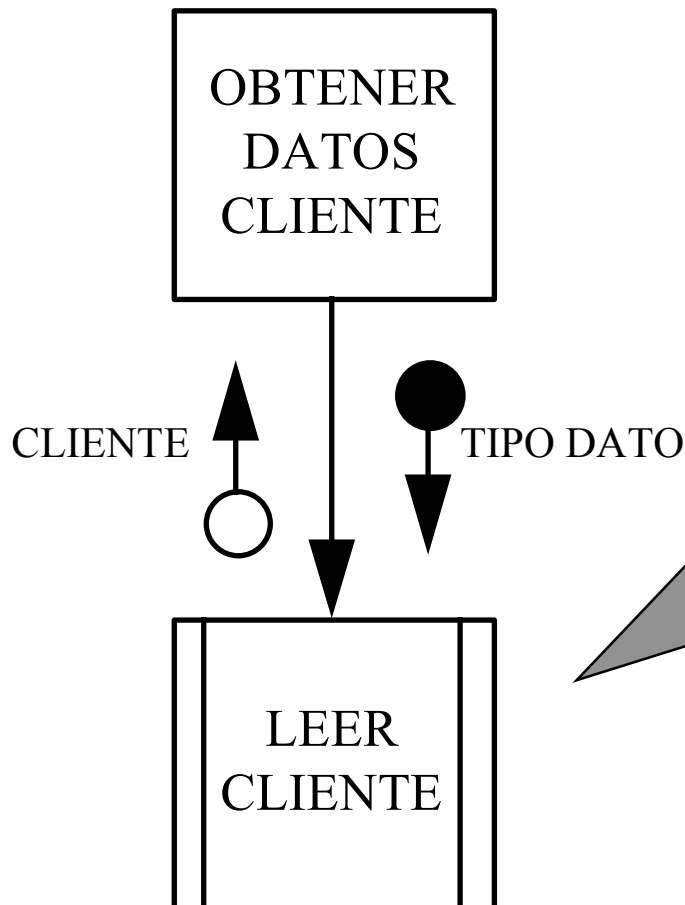
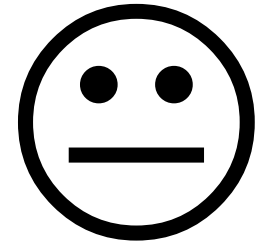
LEER
CLIENTE



Dos módulos se comunican haciendo referencia a la misma estructura de datos (la estructura no es global).
A veces se pasa más información de la necesaria.

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO DE CONTROL



Un módulo pasa a otro elementos de control como argumentos.
Un módulo controla a otro.
En este caso es preferible dividir el módulo en tantos independientes como sea necesario

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO EXTERNO

Dos módulos tienen acoplamiento externo si ambos hacen referencia a **una variable global**, pero las referencias entre módulos consisten en registros individuales de datos y no en la estructura global de datos.



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO COMUN (GLOBAL)

Un grupo de módulos están acoplados comúnmente cuando comparten una estructura global de datos (no solamente una variable global)

(Existe un entorno común).



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

ACOPLAMIENTO POR CONTENIDO

El acoplamiento por contenido es un tipo de acoplamiento patológico. Hay que evitarlo a toda costa.

Dos módulos presentan acoplamiento por contenido si uno hace una referencia al interior del otro (Un módulo modifica algún elemento en otro módulo, un módulo utiliza una variable local de otro, dos módulos comparten los mismos contenidos, etc.).



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

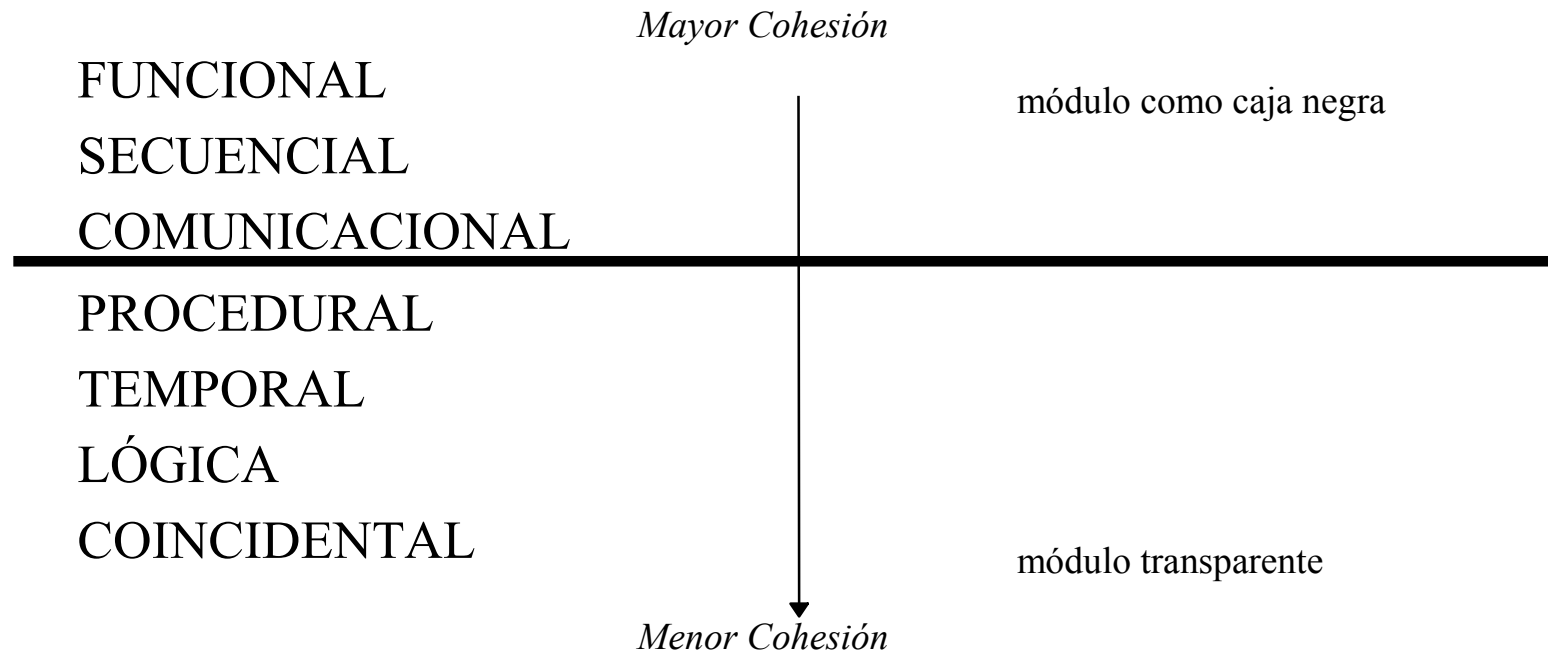
Cohesión

Indica la relación que existe entre los elementos de un mismo módulo. Es la medida de la **relación funcional** de los elementos de un módulo.

El objetivo es organizar estos elementos de manera que los que tengan una mayor relación a la hora de realizar una tarea pertenezcan al mismo módulo, y los elementos no relacionados, se encuentren en módulos separados.

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION FUNCIONAL



Todos los elementos que componen el módulo están relacionados en el desarrollo de una única función

Para reutilizar una de estas funciones no es necesario en absoluto conocer los detalles internos

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION SECUENCIAL 😊😊

Existe cohesión secuencial cuando el módulo representa el empaquetamiento físico de varios módulos con cohesión funcional. Se usa cuando varios módulos con cohesión funcional trabajan secuencialmente, y donde la salida de uno es la entrada del siguiente

Ej. Formatear registro → Primero hace actividades de ‘leer registro’ y posteriormente de ‘formatearlo’

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION COMUNICACIONAL ☺☺

Un módulo con cohesión comunicacional es aquel cuyos elementos o actividades utilizan los mismos datos de entrada y salida. Los módulos con cohesión comunicacional y los que tienen cohesión secuencial parecen similares, ya que contienen actividades organizadas en torno a los datos del problema

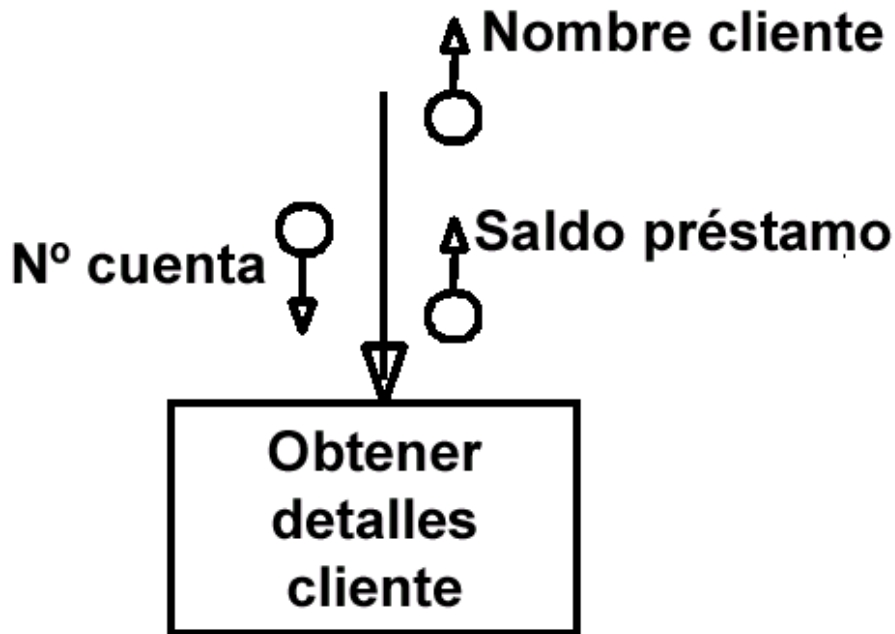
Ej. Leer registro cliente → realiza actividades de leer el nombre y la dirección del cliente. Estas actividades utilizan los mismos datos (los del cliente), y son actividades en las que en principio no importa el orden

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION COMUNICACIONAL



■ ejemplo:



Modulo obtener detalles cliente
usa Numero cuenta
retorna Nombre cliente, saldo préstamo
principio
encontrar nombre cliente
encontrar saldo cliente
fin

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION PROCEDIMENTAL



Este tipo de cohesión se da cuando el módulo tiene una serie de elementos (funciones) relacionados por un procedimiento efectuado por el código.

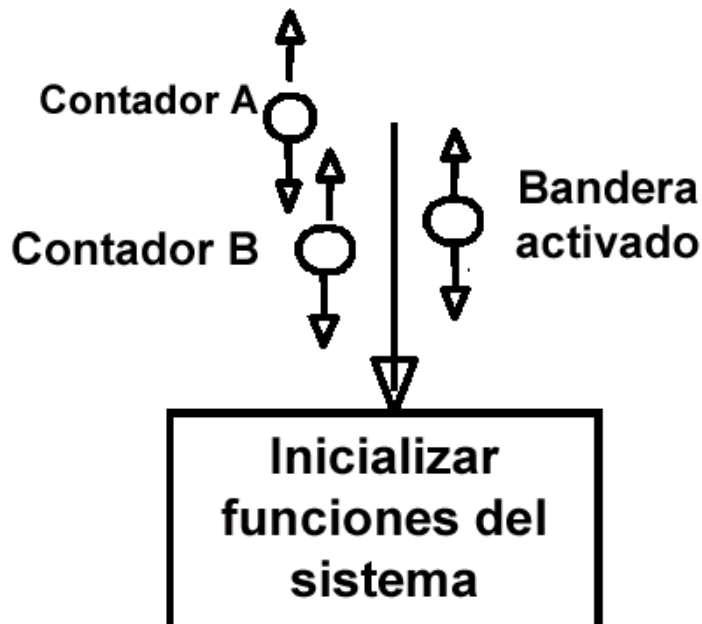
(Cuando un módulo contiene un conjunto de funciones o procedimientos que en principio no tienen ninguna relación unos con otros. Esto provoca que sea muy difícil mantener el módulo, y que sea prácticamente imposible la reutilización).

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION TEMPORAL



Un módulo con cohesión temporal es aquel cuyos elementos están implicados en actividades que están relacionadas en el tiempo



Modulo Inicializar funciones del sistema
usa, retorna contador A, contador B, banderaActivado
principio
rebobinar cinta A
poner contadores A, B a cero
rebobinar cinta B
poner bandera activado a Falso
fin

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION LOGICA



Un módulo tiene cohesión lógica cuando existe alguna relación lógica entre los elementos del módulo.

En algunos casos puede dar lugar a confusiones por no estar bien definidas las fronteras entre los diferentes elementos del módulo.

En cada ejecución tan sólo se ejecutan algunas funciones del módulo, atendiendo a un conjunto de funciones lógicas

ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION LOGICA



**Modulo Rutina general de
Entrada/Salida**

**usa bandera de selección
actualiza registro A
retorna registro B, registro C
principio**

si bandera de selección = 1

.....

Si no si bandera de selección = 2

.....

Si no si bandera de selección = 3

.....

fin



ATRIBUTOS DE LA CALIDAD DE UN DISEÑO

COHESION COINCIDENTAL



Se dice que en un módulo existe cohesión coincidental cuando entre los elementos que lo componen no existe ninguna relación con sentido.

METODOLOGIAS DE DISEÑO DE PROGRAMAS

Permiten conseguir una estructura jerárquica del programa tomando como punto de partida una especificación detallada de la entrada, la salida y los algoritmos del programa a construir.

Las más conocidas son la de Jackson y la de Warnier

METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON



Se basa en el principio de que la base inicial del diseño del programa son los datos del problema y no los requisitos funcionales exigidos.



Permite una mayor objetividad.



Partir de una buena especificación del problema que queremos resolver: datos de entrada, datos de salida y algoritmos aplicables.

Una vez obtenida una estructura objetiva del problema, que constituye un reflejo del mundo real con el que trata el programa, resulta más fácil asignar las distintas funciones a realizar.

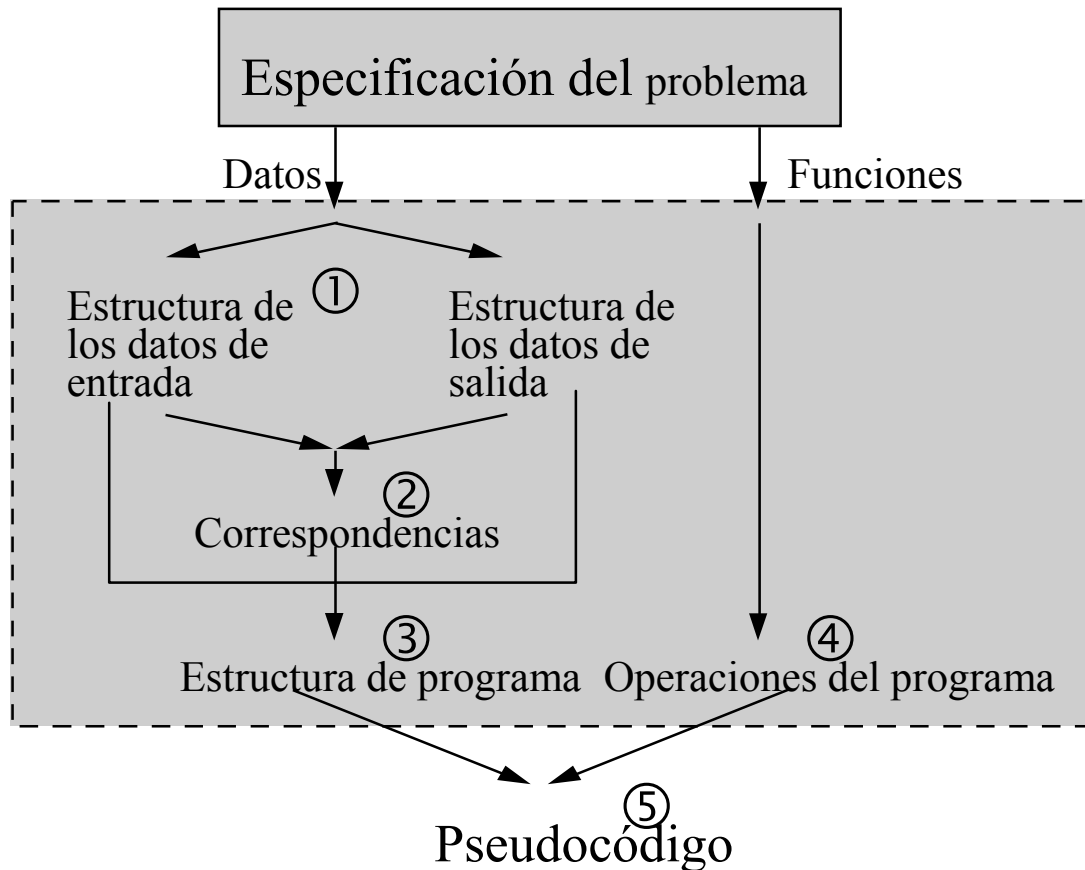
METODOLOGIAS DE DISEÑO DE PROGRAMAS

Fases del Modelo de Jackson

- ☑ Formar las estructuras de datos de salida (estructura lógica de salida) y de entrada (estructura lógica de entrada) a partir de los datos del problema.
- ☑ Determinar las correspondencias (o los elementos comunes) entre ambas estructuras de datos.
- ☑ En función de las correspondencias obtener una estructura única para el programa, que puede traducirse fácilmente a un diagrama de flujo de control.
- ☑ Asignar a la estructura del programa las operaciones ejecutables de programa derivadas de las especificaciones funcionales
- ☑ Traducir el conjunto estructura-operaciones a un formato de pseudocódigo (lógica esquemática) cuya codificación resulta bastante sencilla.

METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON



Las estructuras de datos de entrada y salida y la estructura del programa se documentan mediante **Diagramas de Estructura de Jackson**

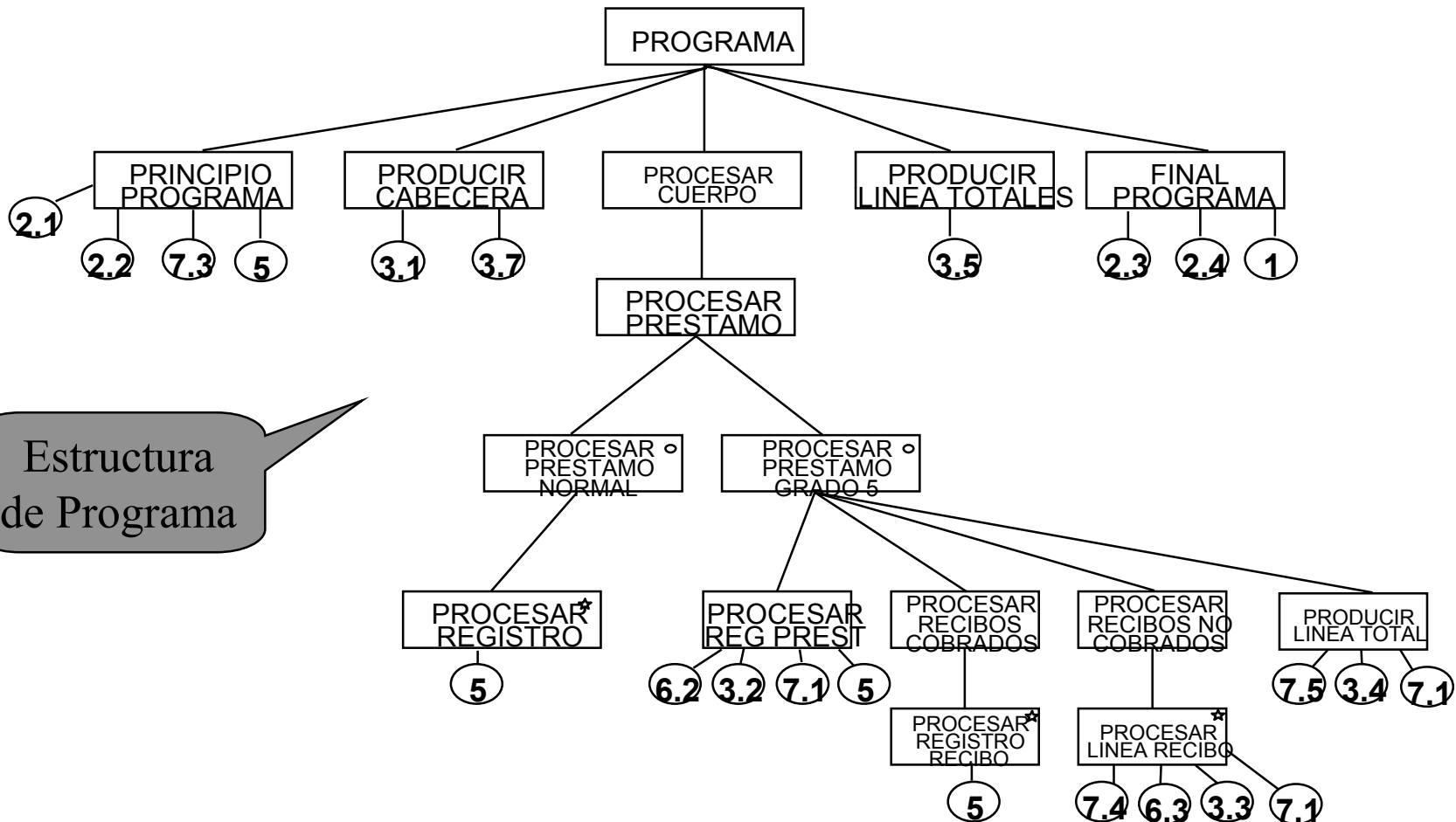
METODOLOGIAS DE DISEÑO DE PROGRAMAS

Diagramas de Estructura de Jackson tiene las siguientes estructuras:

- **SECUENCIA:** Se tiene una estructura de secuencia cuando dos o más componentes son colocados juntos en estricto orden secuencial para formar un componente mayor.
- **REPETICION:** Se usa una construcción de repetición cuando un componente o elemento de datos se repite varias veces. La iteración, a diferencia de las otras estructuras, está formada por un único subcomponente.
- **SELECCION:** La selección se muestra cuando se debe escoger entre dos o más componentes. La selección puede tener una, dos o más de dos alternativas.

METODOLOGIAS DE DISEÑO DE PROGRAMAS

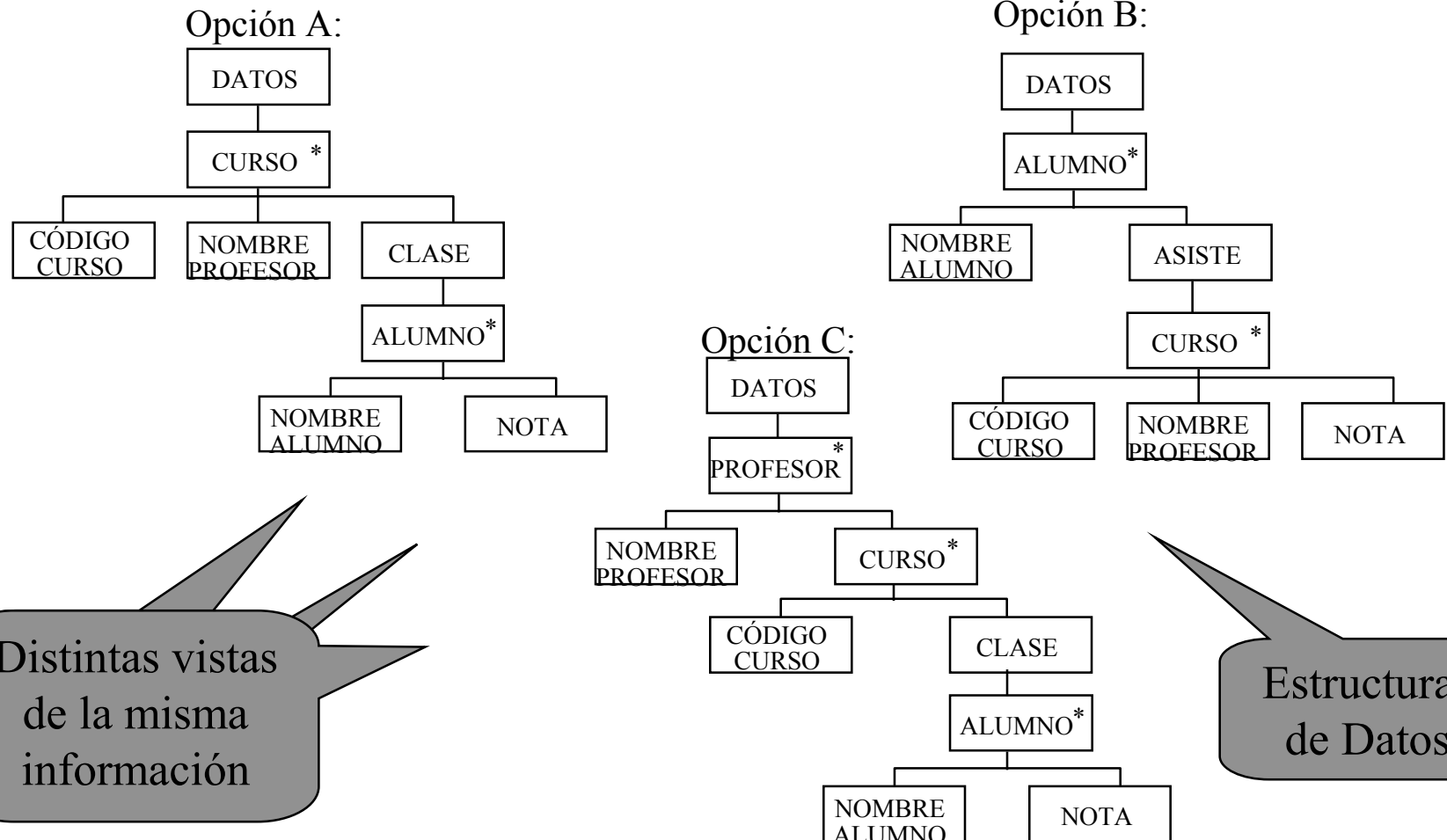
MODELO JACKSON



Estructura de Programa

METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON

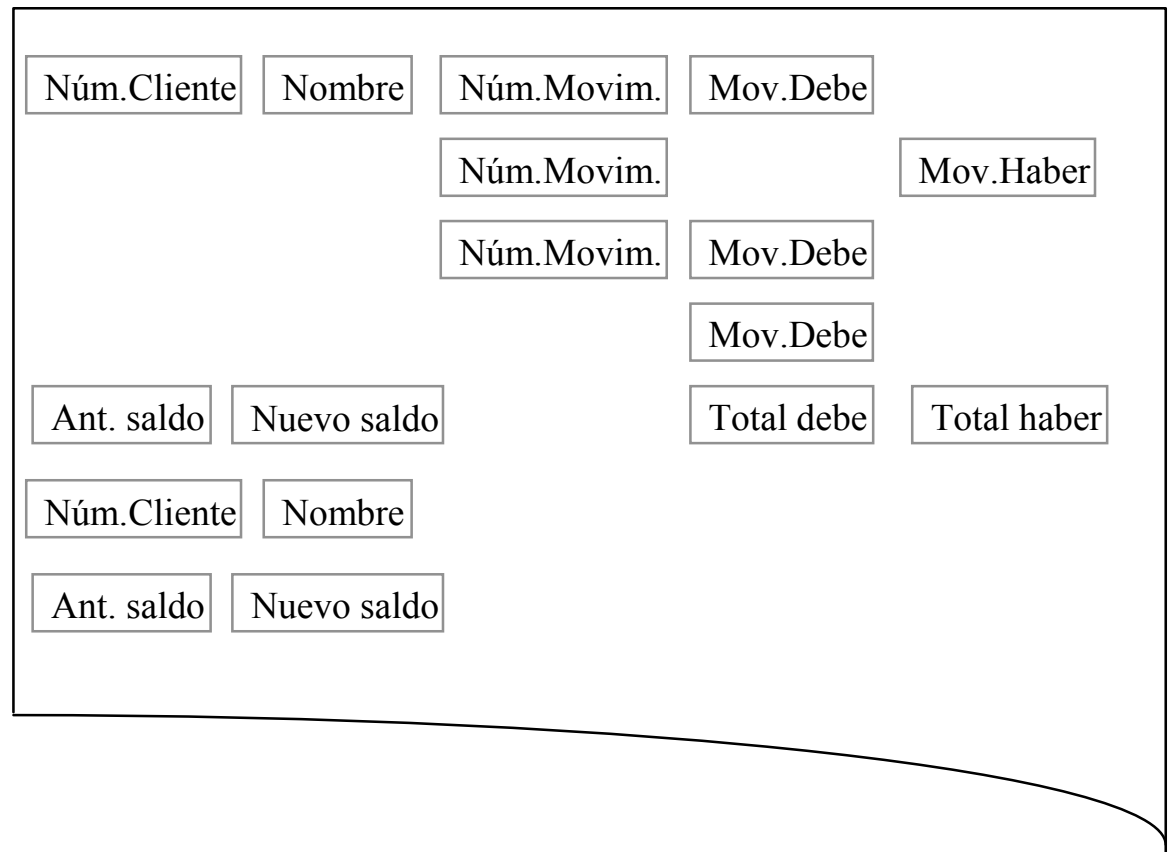


Distintas vistas de la misma información

Estructuras de Datos

METODOLOGIAS DE DISEÑO DE PROGRAMAS**MODELO JACKSON (Ejemplo)**

Diseñar un programa para listar el estado de cuentas de los clientes a partir de la información de saldos y movimientos de los mismos. La salida debe seguir el modelo mostrado a la derecha, calculando el nuevo saldo. Se dispone como entrada de un fichero que contiene dos tipos de registros, uno con los datos y el saldo del cliente, y otro que contiene los movimientos producidos en la cuenta hasta el momento (ordenado ascendentemente por el número de cliente). El archivo puede estar vacío.



METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON

Contenido de los registros del fichero de entrada

Num. Cliente

Nombre

Antiguo saldo

Num. Cliente

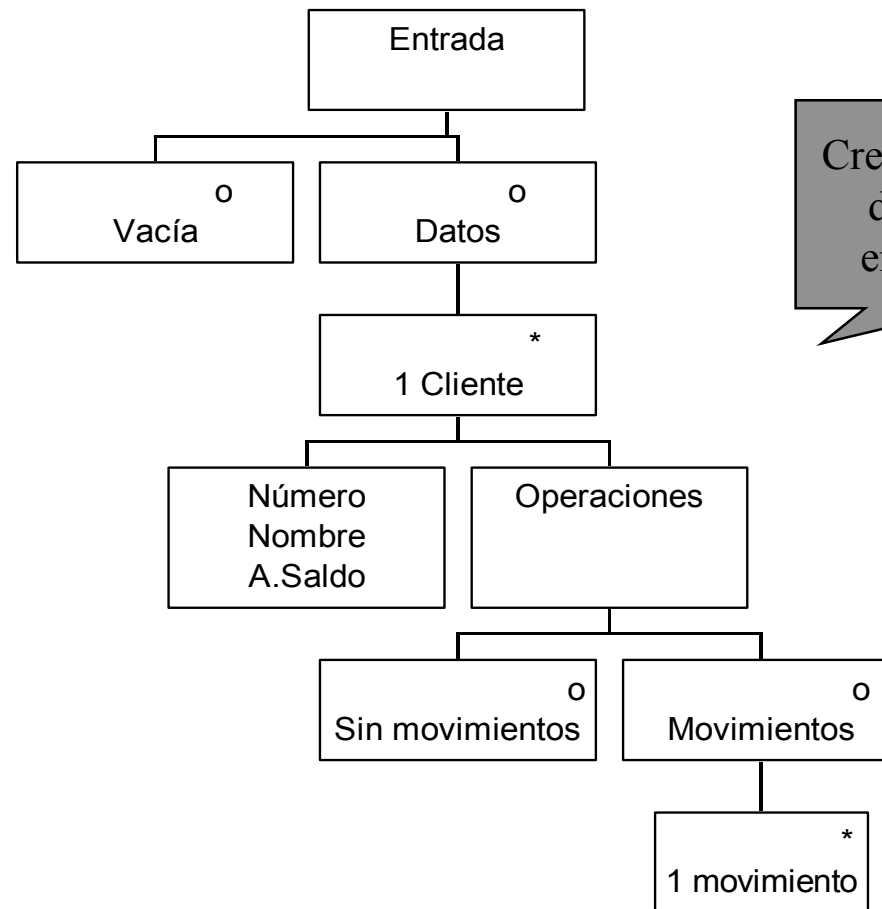
Num.Movimiento

Importe

Código (Debe/Haber)

METODOLOGIAS DE DISEÑO DE PROGRAMAS

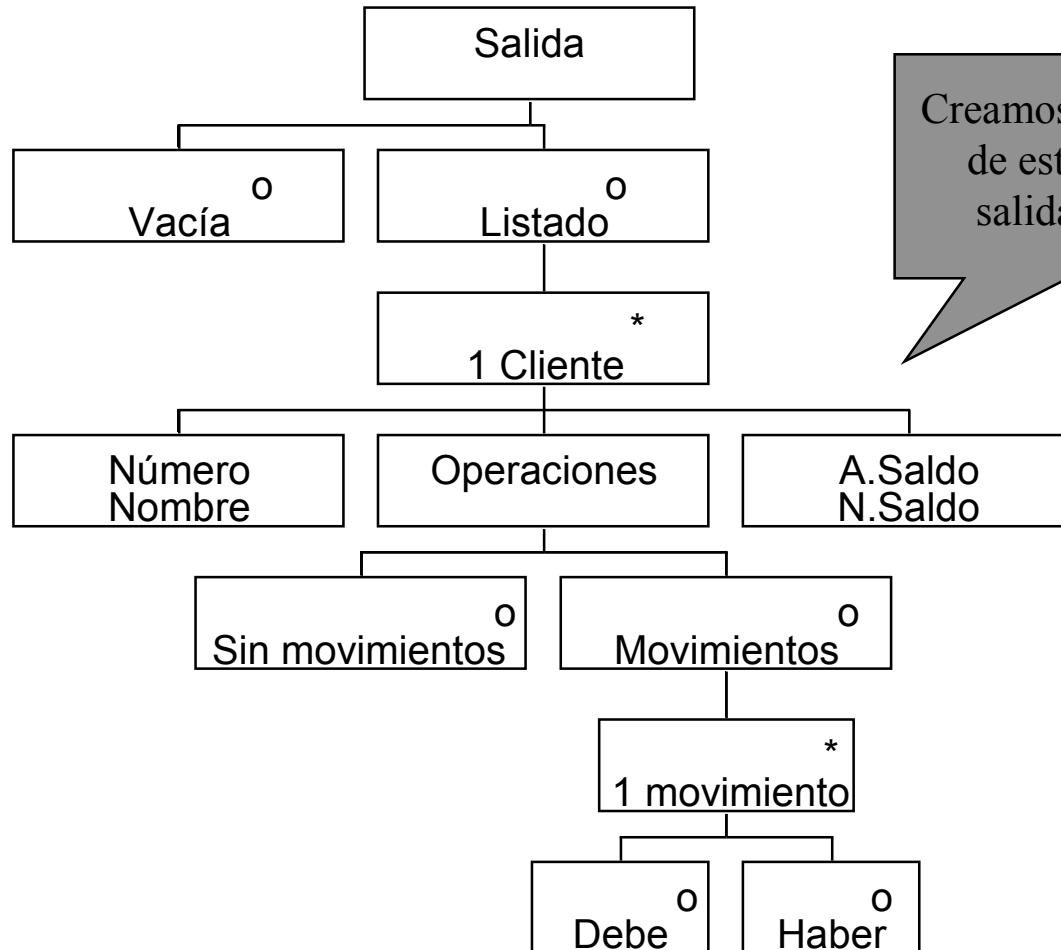
MODELO JACKSON



Creamos el diagrama de estructura de entrada (archivo)

METODOLOGIAS DE DISEÑO DE PROGRAMAS

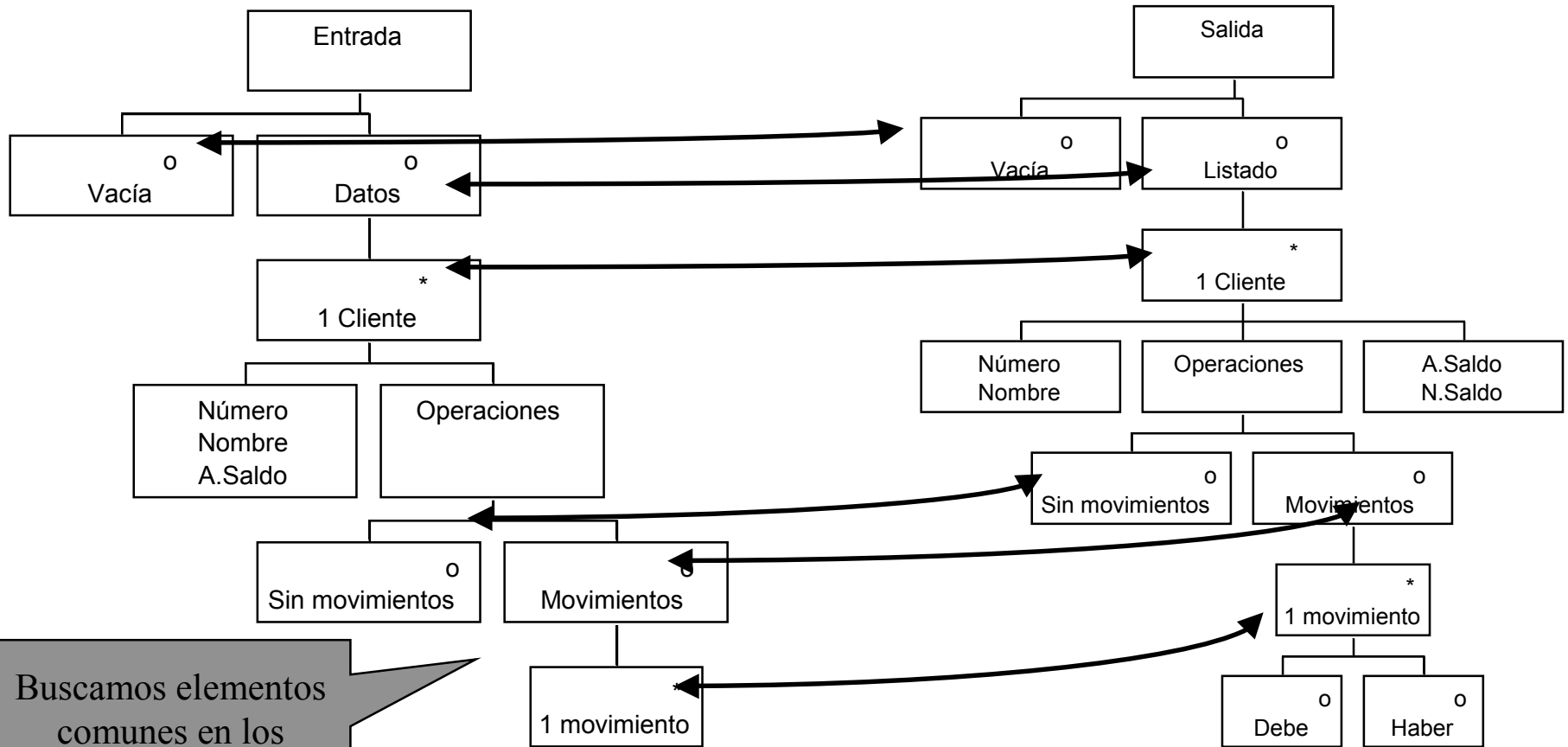
MODELO JACKSON



Creamos el diagrama de estructura de salida (listado)

METODOLOGIAS DE DISEÑO DE PROGRAMAS

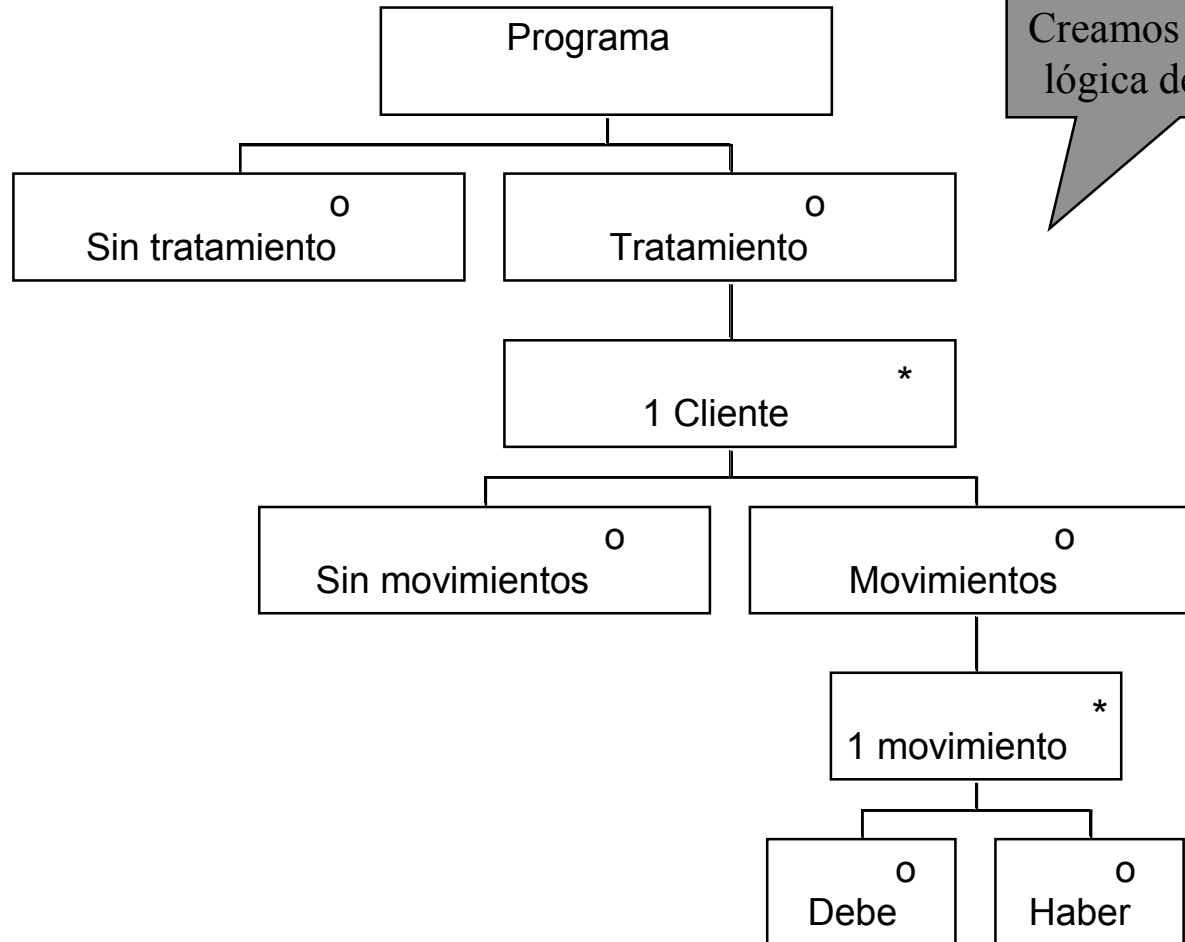
MODELO JACKSON



Buscamos elementos comunes en los modelos de entrada y salida

METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON



Creamos la estructura lógica del programa

METODOLOGIAS DE DISEÑO DE PROGRAMAS

MODELO JACKSON

If entrada no vacía

then

while no fin fichero do

if cliente con movimiento

then

while haya movimientos do

if debe then tratar mov.debe

else tratar mov.haber

else tratamiento sin mov.

total cuenta de cliente;

else tratamiento fichero vacío.

finalizar tratamiento.

METODOLOGIAS DE DISEÑO DE PROGRAMAS

METODOLOGIA WARNIER

Se basa en la aplicación de dos principios:

1. El principio de la ordenación jerárquica de los conjuntos de información (salida, entrada y programa).
2. El principio de correspondencia en la organización de los conjuntos de información

METODOLOGIAS DE DISEÑO DE PROGRAMAS

METODOLOGIA WARNIER

- ◆ Secuencia, representada por diversos elementos que se suceden de arriba a abajo en un mismo nivel.
- ◆ Repetición de ocurrencias dentro de un mismo conjunto, que se representan en los diagramas indicando el número mínimo y máximo de las mismas: por ejemplo, (0,n).
- ◆ Selección entre ocurrencias de un conjunto, se efectúa la subdivisión en subconjuntos cuya presencia es aleatoria y excluyentes entre sí, y se representa por medio del símbolo +

METODOLOGIAS DE DISEÑO DE PROGRAMAS

METODOLOGIA WARNIER

Principio de Correspondencia

- ⊗ La organización jerárquica de los datos de entrada determinada por los datos de salida
- ⊗ La organización del programa viene determinada por los datos de entrada
- ⊗ El control del programa se realiza a partir de los datos de salida

METODOLOGIAS DE DISEÑO DE PROGRAMAS

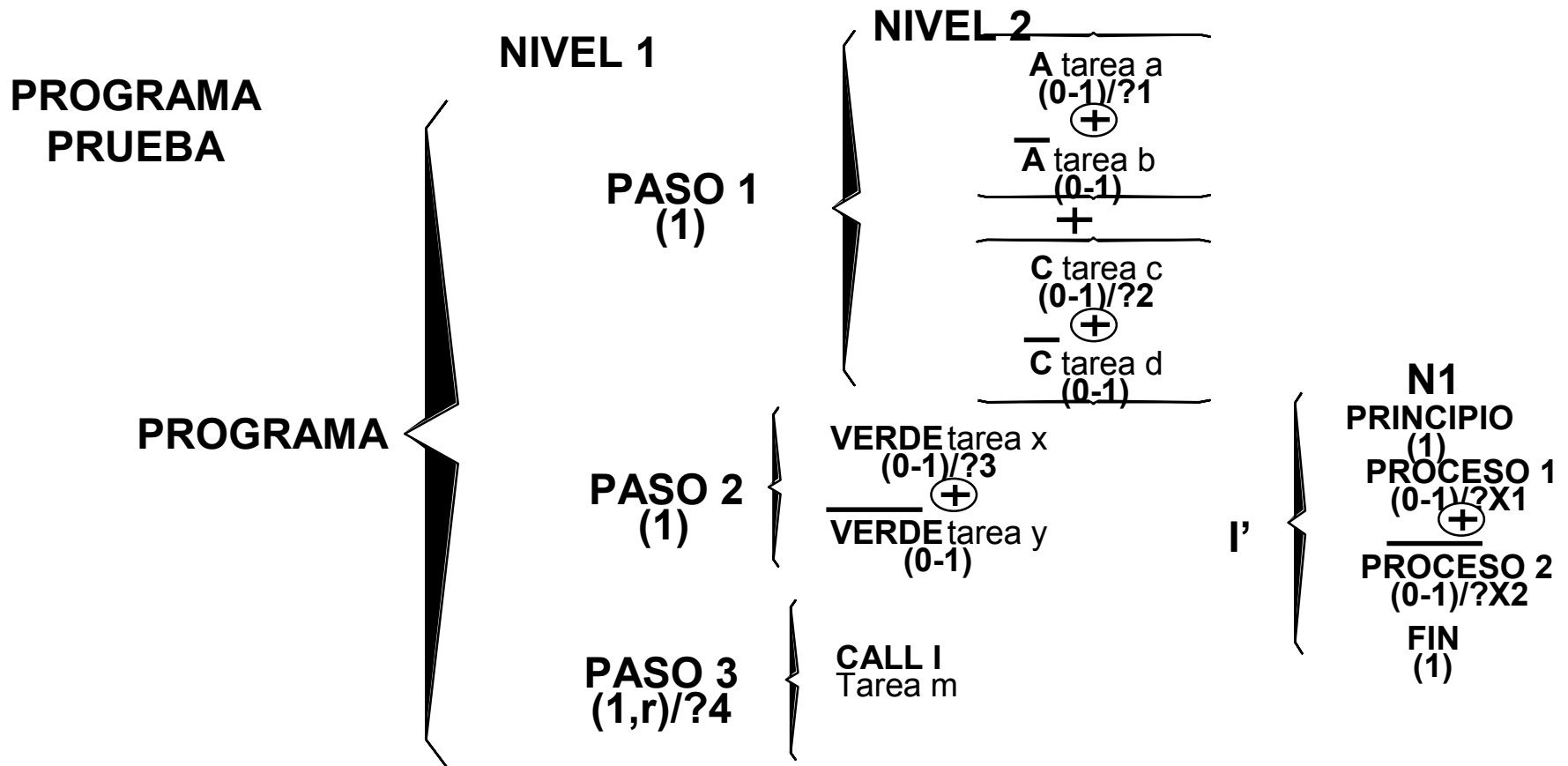
METODOLOGIA WARNIER

Fases de la Metodología

- ☞ Determinación de la estructura de los datos de salida
- ☞ Determinación de la estructura de los datos de entrada en función de la salida deseada
- ☞ Determinación de la estructura óptima del programa basada en el estructura de entrada
- ☞ Creación de una lista de pseudoinstrucciones
- ☞ Asignación de las mismas a cada elemento de la estructura del programa

METODOLOGIAS DE DISEÑO DE PROGRAMAS

METODOLOGIA WARNIER



Ejercicio 1

Una empresa compra a una serie de proveedores diferentes piezas que posteriormente venderá a sus clientes, debiendo llevar a cabo el control de almacén (nº de piezas existentes de cada una de ellas).

La aplicación debe gestionar los proveedores, así como las piezas que proporciona cada uno (proveedor y piezas con sus respectivos precios, corresponde al flujo de entrada «proveedor»). Con los proveedores y las piezas que proporciona cada uno de ellos, se genera una lista de precios que se corresponde con los precios que consideremos mejores para cada una de las piezas que se puedan proporcionar al cliente (como criterio de selección se encuentra entre otros la marca de la pieza).

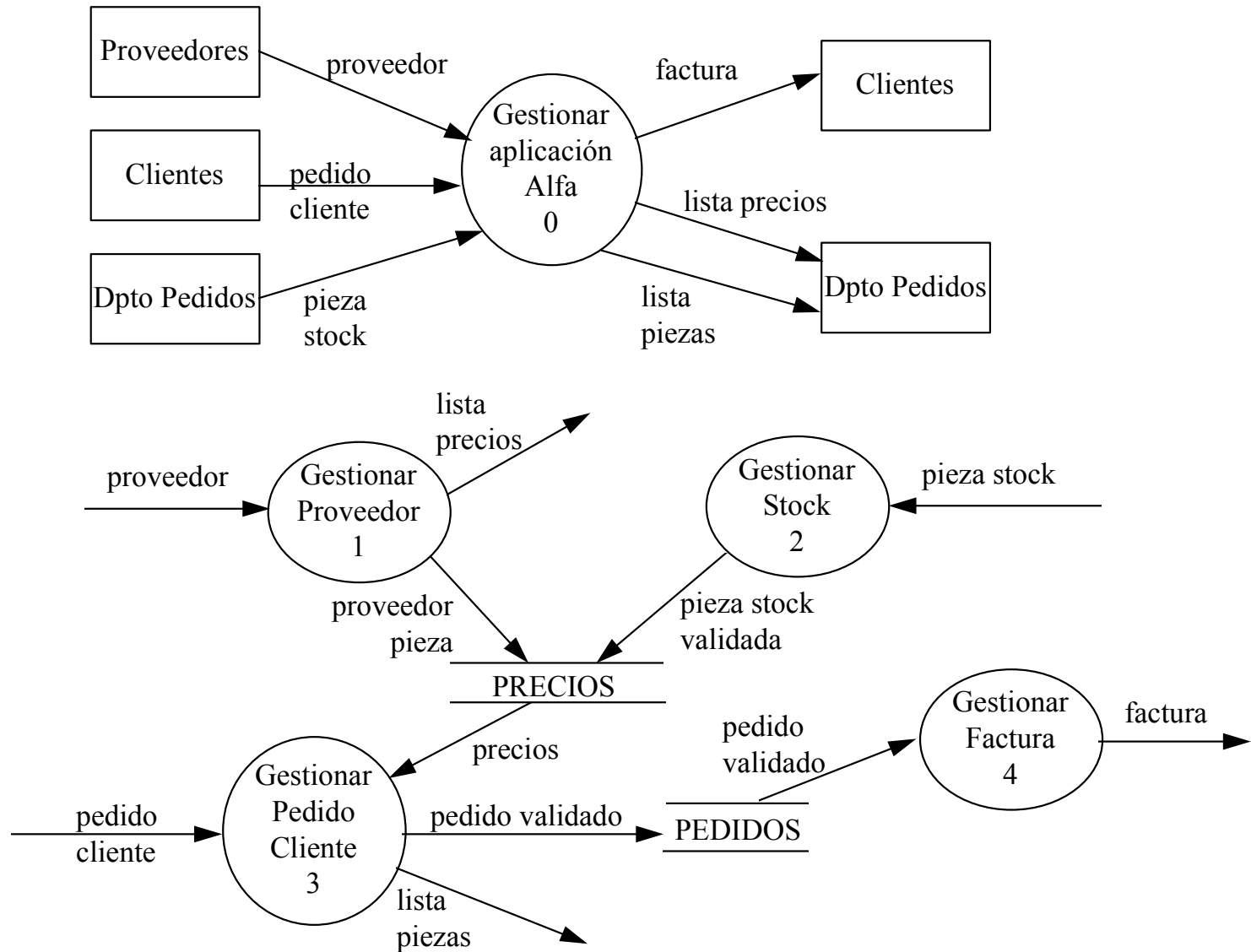
El control del almacén, es decir, las cantidades que tenemos de las diferentes piezas que hemos pedido a los proveedores (flujo de datos de «pieza stock»), determinará si el pedido realizado por el cliente («pedido cliente») se puede satisfacer completamente o no, según tengamos o no las piezas pedidas (generando en el caso de no tener dichas piezas un listado de ellas, «lista piezas»).

Cuando el pedido se entrega al cliente, se genera la factura correspondiente.

Cada una de estas funciones (en el DFD 1, 2, 3 y 4) puede realizarse en cualquier momento, independientemente de las demás funciones. Se pide dibujar el diagrama de estructuras correspondiente indicando si la característica principal del DFD es de transformación o transacción.

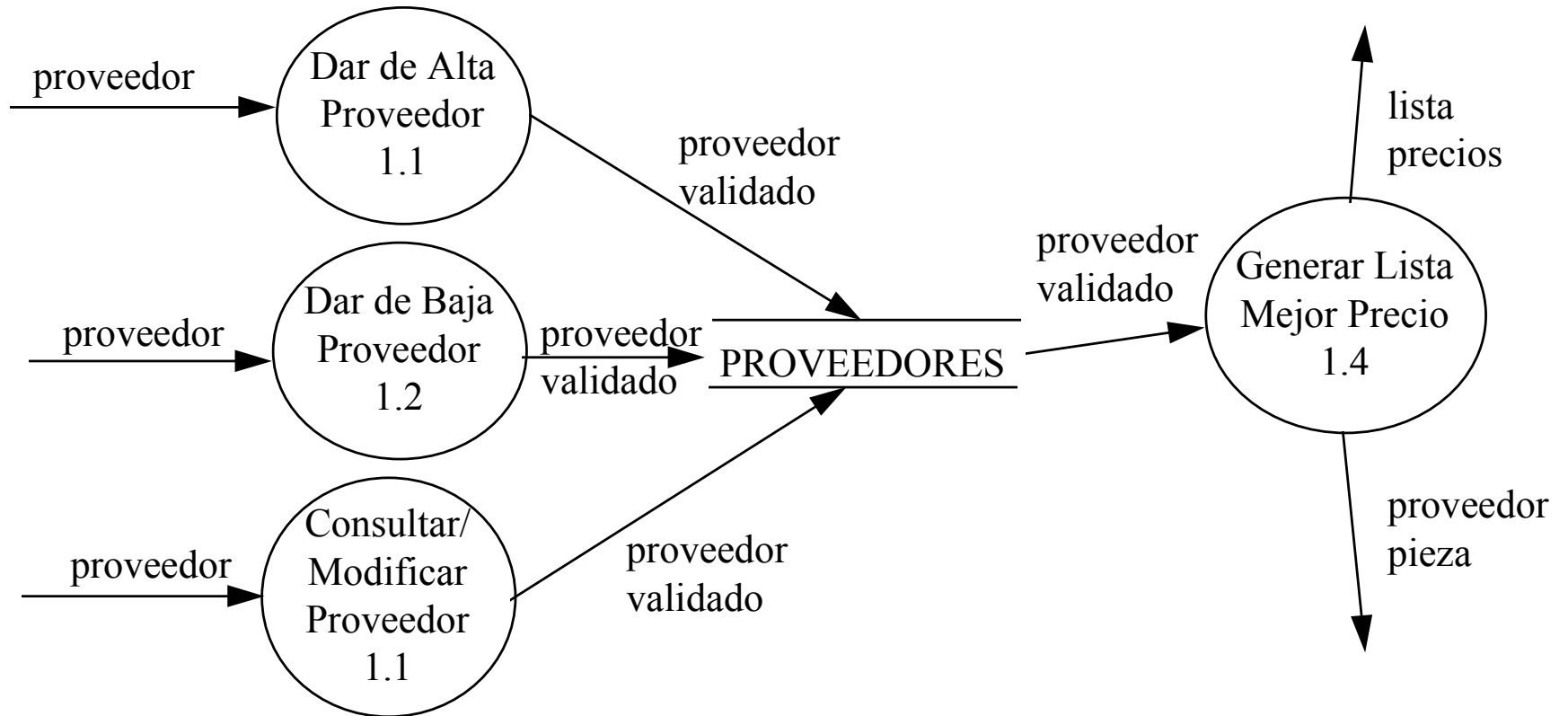
DISEÑO ESTRUCTURADO DE SISTEMAS

8.010



DISEÑO ESTRUCTURADO DE SISTEMAS

8.010



Ejercicio 2

Se desea automatizar la gestión de un VideoClub. El funcionamiento del sistema es el siguiente:

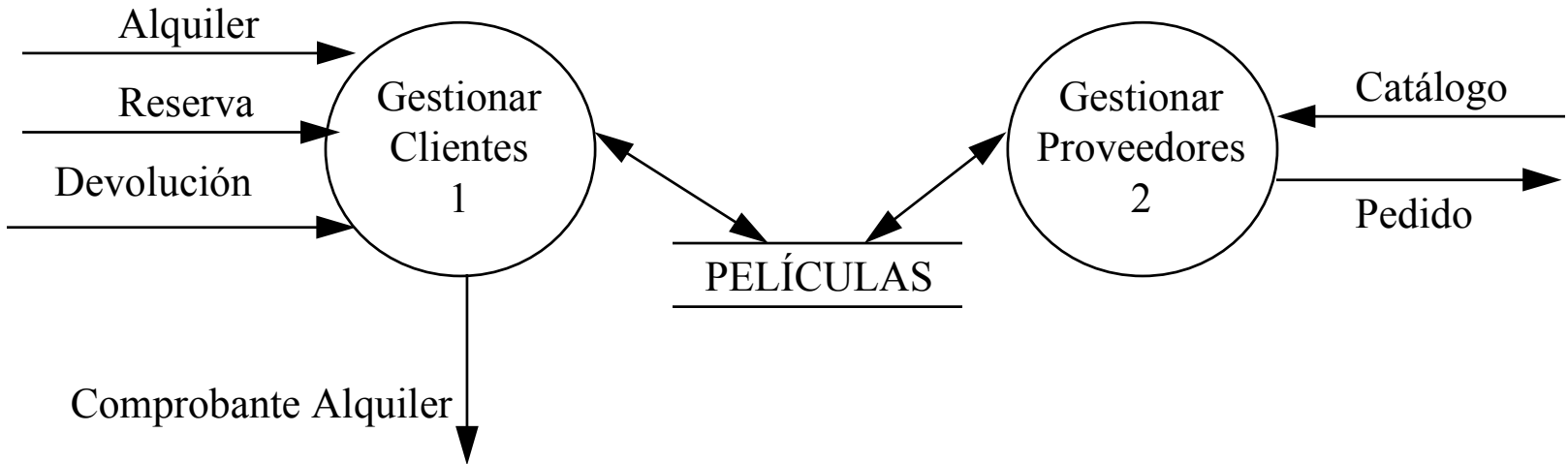
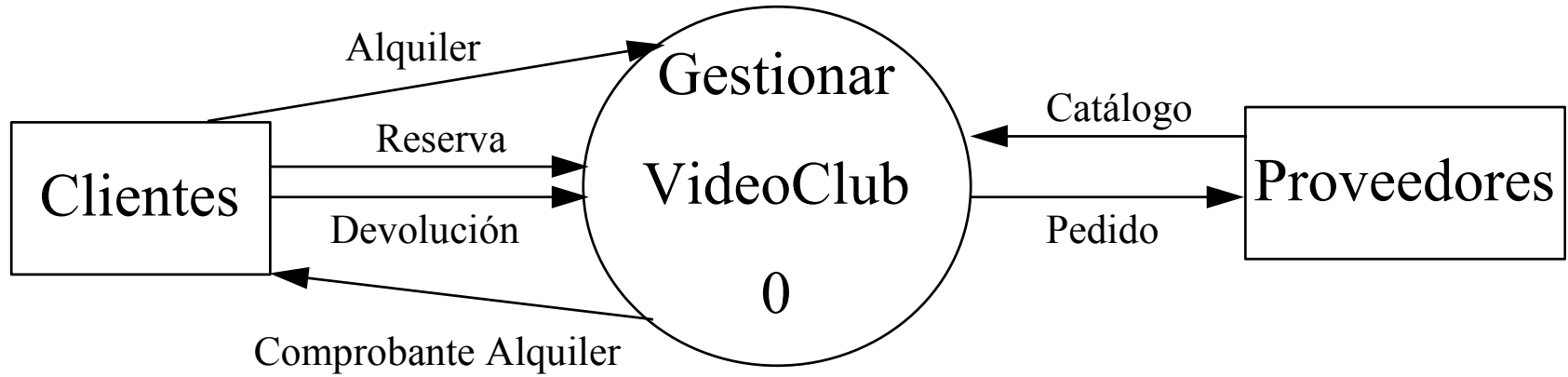
- Por un lado, es necesario tratar a los proveedores y, por otro, es necesario tratar a los clientes.
- El tratamiento de los proveedores incluye, en cualquier instante y de forma independiente, la realización del registro de los catálogos de películas y la generación de pedidos de películas al proveedor.
- Cuando llega un cliente al videoclub, éste solicita el tipo de gestión que quiere realizar, es decir, alquiler, reserva o devolución de película. Cuando quiere realizar un alquiler de una película, el proceso, con la información del alquiler, comprobará si existe stock suficiente de esa película, así como reserva. En caso de ser satisfactorias estas comprobaciones (existe la película y no está reservada por otro cliente), se disminuirá el stock de esa película y se registrará el alquiler generando un comprobante de alquiler para el cliente. Cuando quiere realizar una devolución de una película, lo que se hace es comprobar que la película estaba alquilada por él y aumentar el stock de esa película. Cuando quiere reservar una película, se registra la reserva de la película.

No se tiene en cuenta el tratamiento de los errores que pudieran ocurrir.

Se pide: realizar el diagrama de estructura correspondiente, indicando si se trata de una transformación (decir cuál es su centro) o de una transacción.

DISEÑO ESTRUCTURADO DE SISTEMAS

8.010



DISEÑO ESTRUCTURADO DE SISTEMAS

8.010

