

# Ingeniería del Software I

2 – Análisis y especificación de los requisitos del software  
(Capítulo 3)

# Análisis y especificación de los requisitos del software

- El problema de escala es fundamental en IS.
- En pequeños problemas:  
comprender y especificar los requerimientos es fácil.
- En grandes problemas:  
comprender y especificar los requerimientos es muy difícil, grandes posibilidades de error.

Entrada:

Las necesidades se encuentran en la cabeza de alguien (ideas abstractas).

Salida:

Un detalle preciso de lo que será el sistema futuro.

# Análisis y especificación de los requisitos del software

- **Identificar y especificar** los requisitos necesariamente involucra interacción con la gente.
- No puede automatizarse.
- La fase de requisitos finaliza produciendo el documento con la especificación de los requerimientos del software (SRS).
- La SRS especifica lo que el sistema propuesto debe hacer.

# Requerimientos del software

# Por qué la SRS es necesaria?

La SRS establece las bases para el **acuerdo** entre el cliente/usuario y quien suministrará el software.

- Hay 3 partes involucradas:  
necesidades del cliente  
consideraciones del usuario } deben comunicarse al desarrollador
  - **Brecha comunicacional** entre las partes:  
Cliente: no comprende el proceso de desarrollo de software.  
Desarrollador: no conoce el problema del cliente ni su área de aplicación.

La SRS es el medio para reconciliar las diferencias y especificar las necesidades del cliente/usuario de manera que todos entiendan.

# Requerimientos del software

Requerimientos (IEEE):

1. Una condición o capacidad necesaria de un usuario para solucionar un problema o alcanzar los objetivos.
2. Una condición o capacidad necesaria que debe poseer o cumplir un sistema [...].

Entender/discernir los requerimientos es complejo:

- Visualizar un futuro sistema es difícil.
- Las capacidades del futuro sistema no están claras, por lo tanto, sus necesidades tampoco.
- Los requerimientos cambian con el tiempo.

Es necesario realizar apropiadamente el análisis y especificación de los requerimientos (SRS: qué y no cómo).

# Requerimientos del software

Por qué la SRS es necesaria?

- Ayuda al usuario a comprender sus necesidades.
- Los usuarios no siempre saben lo que quieren o necesitan.  
Debe analizar y comprender el potencial.
- El proceso de requerimientos ayuda a aclarar las necesidades.
- La SRS provee una referencia para la validación del producto final.
- Debería dar una clara comprensión de lo que se espera.

Objetivo: no sólo automatizar un sistema manual, sino también  
agregar valor a través de la tecnología.

Verificación: “el software satisface la SRS”.

# Requerimientos del software

Por qué la SRS es necesaria?

Una SRS de alta calidad es esencial para obtener un software de calidad.

- Los errores de requerimientos sólo se manifestarán en el software final.
- Para satisfacer los objetivos de calidad, se debe comenzar con una SRS de calidad.
- Los defectos de requerimientos no son pocos. Ej.:
  - 45% de los errores en testing correspondían a errores de requerimientos (siendo el 25% del total de defectos encontrados en el proyecto).
  - 80 errores del proyecto A-7, resultaron en solicitud de cambios.
  - 500 y 250 defectos en dos SRS previamente aprobadas.

# Requerimientos del software

## Por qué la SRS es necesaria?

Una buena SRS reduce los costos de desarrollo.

- Los errores en SRS son más caros de corregir a medida que progresá el proyecto.
- Los cambios de requerimientos pueden costar demasiado (hasta un 40%).
- Una buena SRS contribuye a minimizar cambios y errores.
- Ahorro sustancial: esfuerzo extra invertido en requerimientos produce ahorros, varias veces, mayores de esfuerzo total.

Phase	Cost (person-hours)
Requirements	2
Design	5
Coding	15
Acceptance test	50
Operation and maint.	150

# Requerimientos del software

## Por qué la SRS es necesaria?

Si se invierten  
100 hs/hombre extras en req.  
para eliminar los 50 errores  
=> reducción neta de 1052  
hs/hombre

Ej.: 65% errores req.  
detectado en diseño, 2% en  
codificación, 30% en testing, 3% en  
operación. Si 50 errores introducidos  
en req. no se eliminaron en req. =>  
Costo total =  $32.5*5 + 1*15 + 15*50 +$   
 $1.5*150 = 1152$  hs

Phase	Cost (person-hours)
Requirements	2
Design	5
Coding	15
Acceptance test	50
Operation and maint.	150

# Requerimientos del software

## Proceso de requerimientos

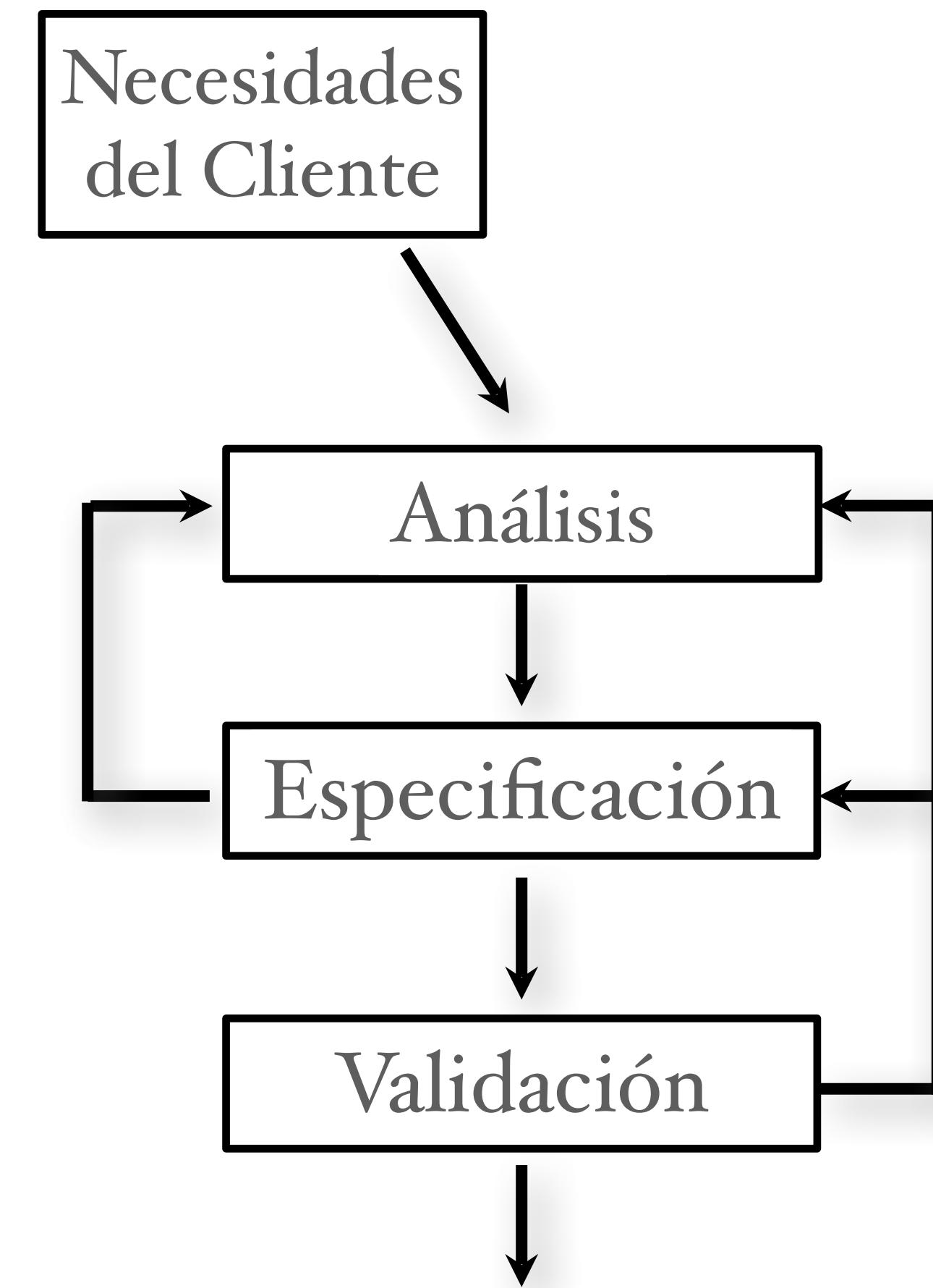
Secuencia de pasos que se necesita realizar para convertir las necesidades del usuario en la SRS.

- El proceso tiene que recolectar las necesidades y los requerimientos y especificarlos claramente.
- Actividades básicas:
  - 1) Análisis del problema o requerimientos.
  - 2) Especificación de los requerimientos.
  - 3) Validación.
- El análisis exige la recolección/extracción y es lo más difícil.

# Requerimientos del software

## Proceso de requerimientos

- El proceso no es lineal; es iterativo y en paralelo.
- Existe superposición entre las fases: algunas partes pueden estar siendo especificadas mientras otras están aún bajo análisis.
- La especificación misma puede ayudar al análisis.
- La validación puede mostrar brechas que conducirán a más análisis y más especificación.



# Requerimientos del software

## Proceso de requerimientos

- El análisis se enfoca en la comprensión del sistema deseado y sus requerimientos.
- Estrategia básica: Dividir y conquistar.  
Descomponer el problema en pequeñas partes; comprender cada una de estas partes y las relaciones entre ellas.
- Se genera una gran cantidad de información.  
La clave es organizarla.  
Durante análisis usar técnicas como diagramas de flujo de datos, diagramas de objetos, etcétera.

# Requerimientos del software

## Proceso de requerimientos

La transición del análisis a la especificación es complicada.

- La especificación se enfoca en el comportamiento externo.
- Objetivo del análisis: comprender la estructura del problema y su dominio: componentes, entrada, salida.
- Se recolecta más información, o distinta, de la necesaria para la especificación.

El uso del análisis y las estructuras que se construyen puede ser indirecto, ayudando a **comprender en lugar de asistir** a la especificación.

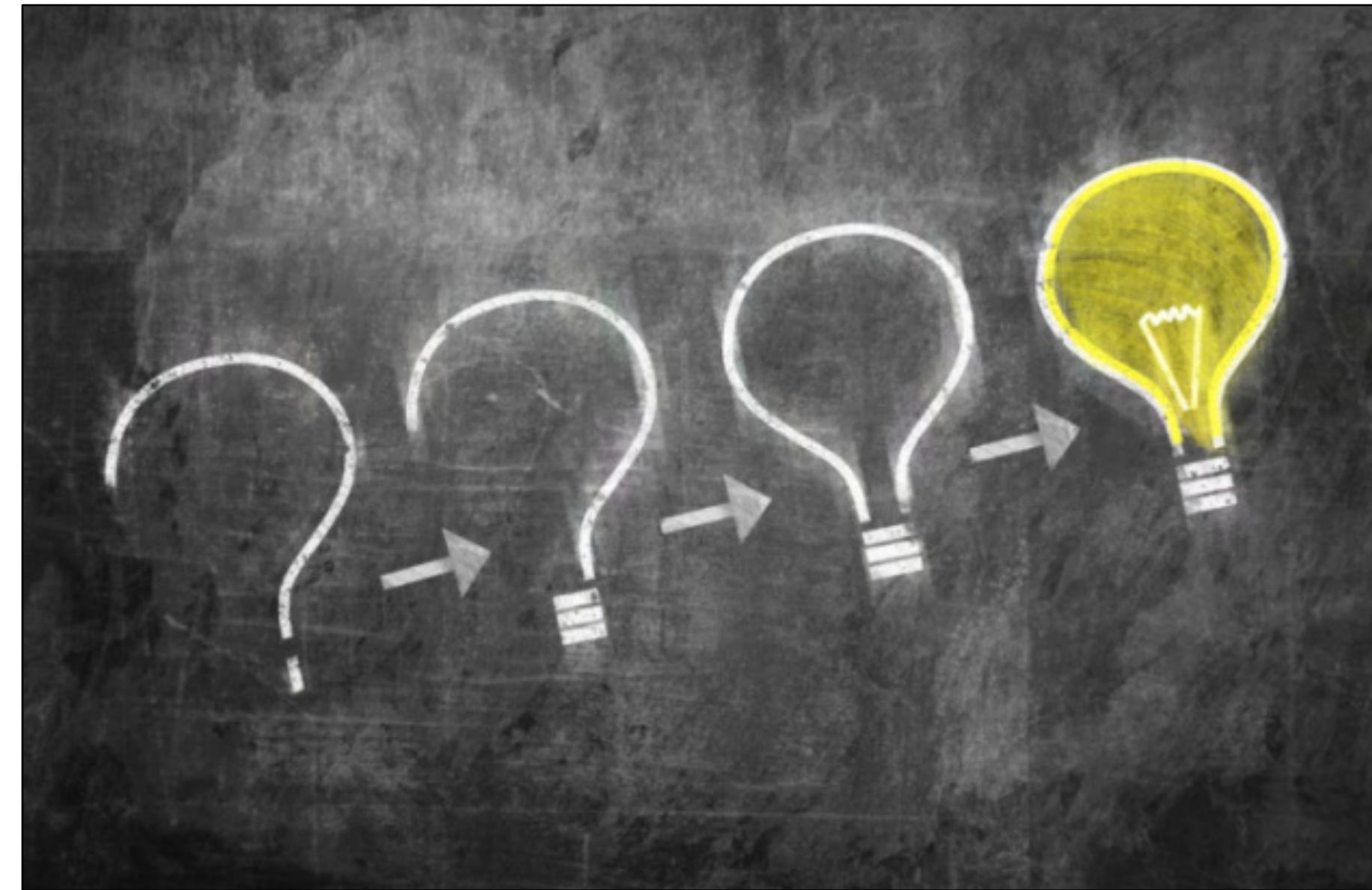
Los métodos de análisis son similares a los de diseño, pero con objetivos y alcances distintos.

El análisis trata con el dominio del problema mientras que el diseño trata con el dominio de la solución.

# Requerimientos del software

Proceso de requerimientos

Formular bien la pregunta ayuda a encontrar la respuesta



# Análisis del problema

Objetivo: lograr una buena comprensión de las necesidades, requerimientos, y restricciones del software.

El análisis incluye:

- entrevistas con el cliente y usuarios,
- lectura de manuales,
- estudio del sistema actual,
- ayudar al cliente/usuario a comprender nuevas posibilidades.

El analista no solo recolecta y organiza la información, i.e. rol pasivo, sino también actúa como consultor, i.e. rol activo.

Debe comprender el funcionamiento de la organización, el cliente, y los usuarios.

# Análisis del problema

Algunas cuestiones:

- Obtener la información necesaria.
- Brainstroming: interactuar con el cliente para establecer las propiedades deseadas.
- Las **relaciones interpersonales** son importantes.
- Habilidad en la comunicación es muy importante.
- Organizar la información dado que se recolecta grandes cantidades.
- Asegurar completitud.
- Asegurar consistencia.
- Evitar diseño interno.

# Análisis del problema

Principio básico: particionar el problema.

Luego comprender cada subproblema y la relación entre ellos, pero... ¿con respecto a qué?

- Funciones: análisis estructural
- Objetos: análisis OO
- Eventos del sistema: particionado de eventos

Proyección: obtener distintas vistas desde distintos puntos de vistas.

# Análisis del problema

## Enfoque informal

- No hay una metodología definida; la información se obtiene a través de análisis, observación, interacción, discusión, ...
- No se construye un modelo formal del sistema.
- La información recogida se plasma y organiza directamente en la SRS, la cual es el objeto de revisión con el cliente.
- Depende de la experiencia del analista y el feedback del cliente en las revisiones.
- Útil en muchos contextos.

# Análisis del problema

## Modelado de flujo de datos

- Ampliamente utilizado.
- Se enfoca en las funciones realizadas en el sistema, no en los requisitos no-funcionales.
- Ve al sistema como una red de transformadores de datos sobre la cual fluye la información.
- Para el modelado utiliza diagramas de flujo de datos (DFD) y descomposición funcional.

La metodología de análisis y especificación estructurada utiliza DFD para organizar la información y guiar el análisis.

# Análisis del problema

## Modelado de flujo de datos: DFD

Un DFD es una representación gráfica para representar un algoritmo/proceso, donde se representa el flujo de datos a través del sistema.

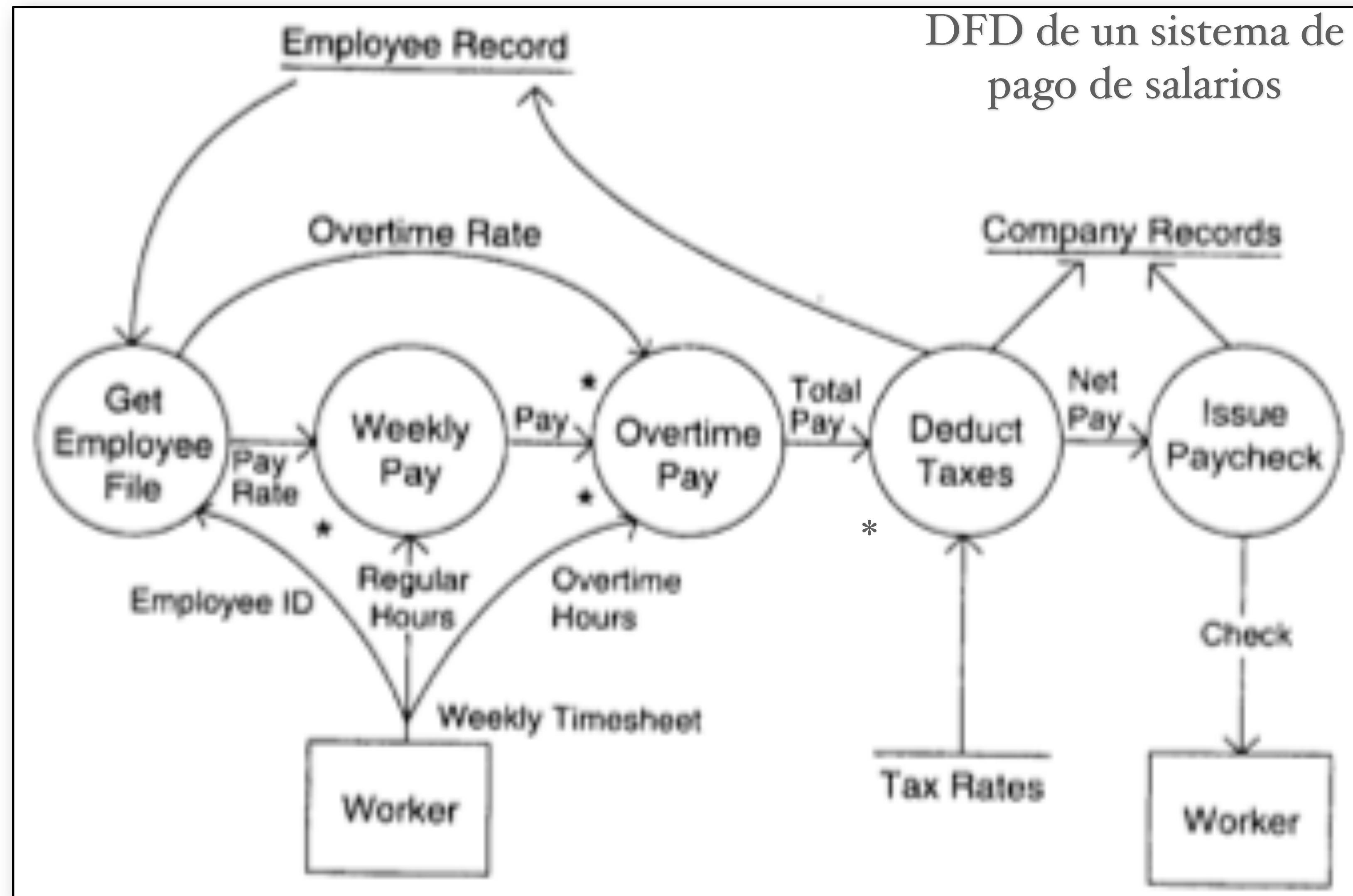
- Ve al sistema como una transformación de entradas en salidas.
- La transformación se realiza a través de “transformadores/procesos”.
- El DFD captura la manera en que ocurre la transformación de la entrada en la salida a medida que los datos se mueven a través de los transformadores.
- No se limita al software.

Es un gráfico lógico del plan de trabajo que se ejecutará para la solución de un determinado problema.

# Análisis del problema

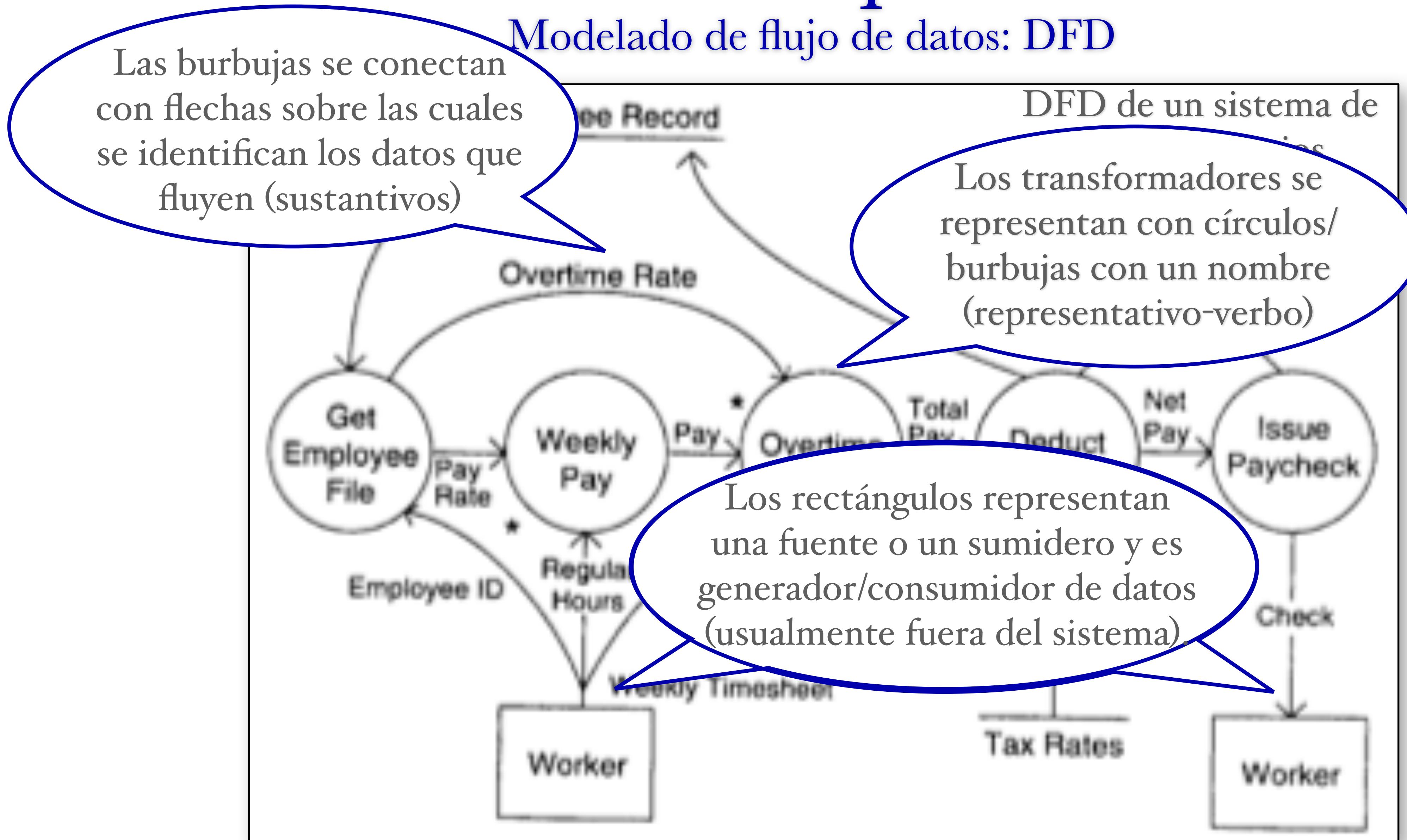
Modelado de flujo de datos: DFD

DFD de un sistema de pago de salarios



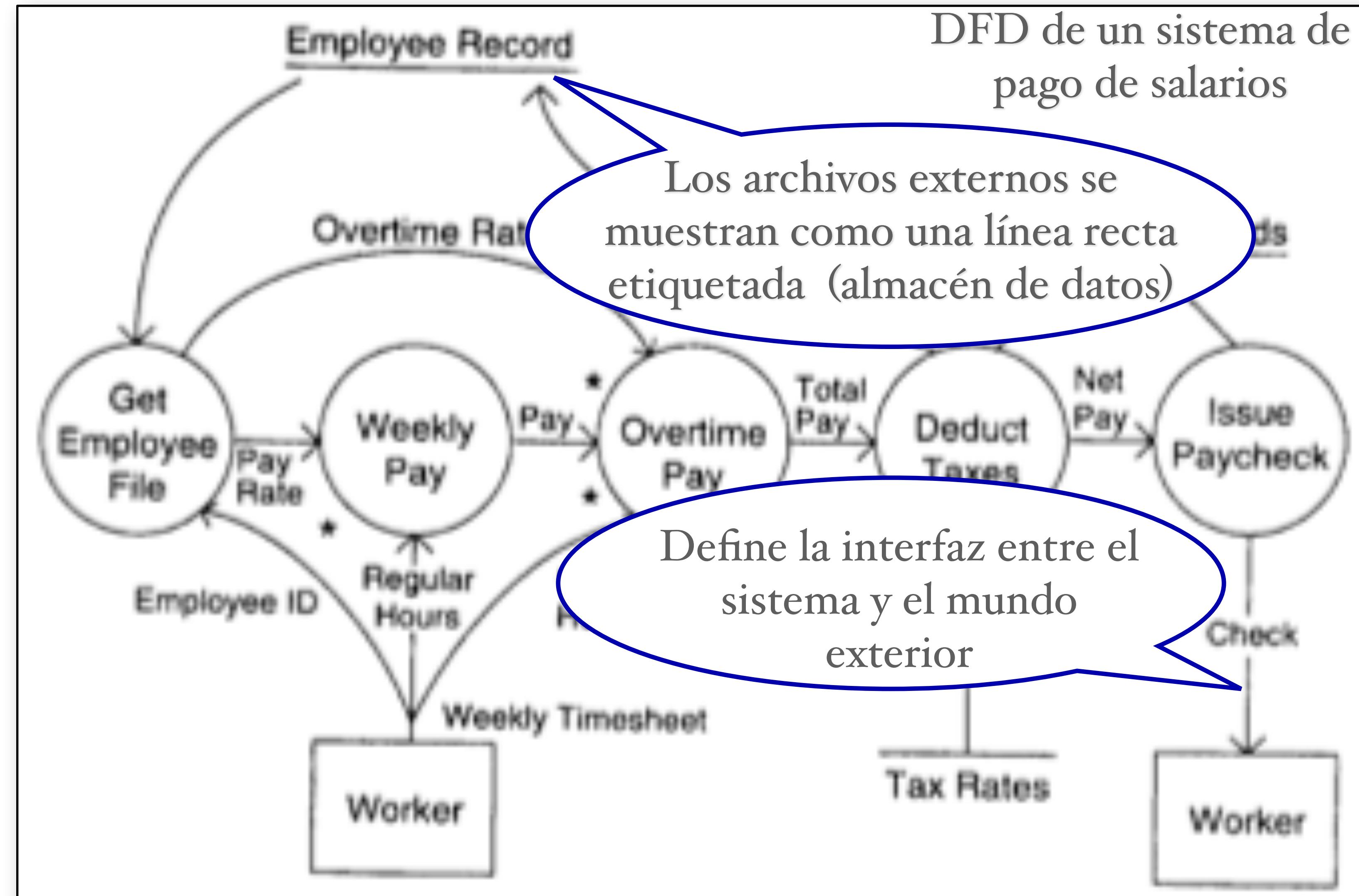
# Análisis del problema

## Modelado de flujo de datos: DFD



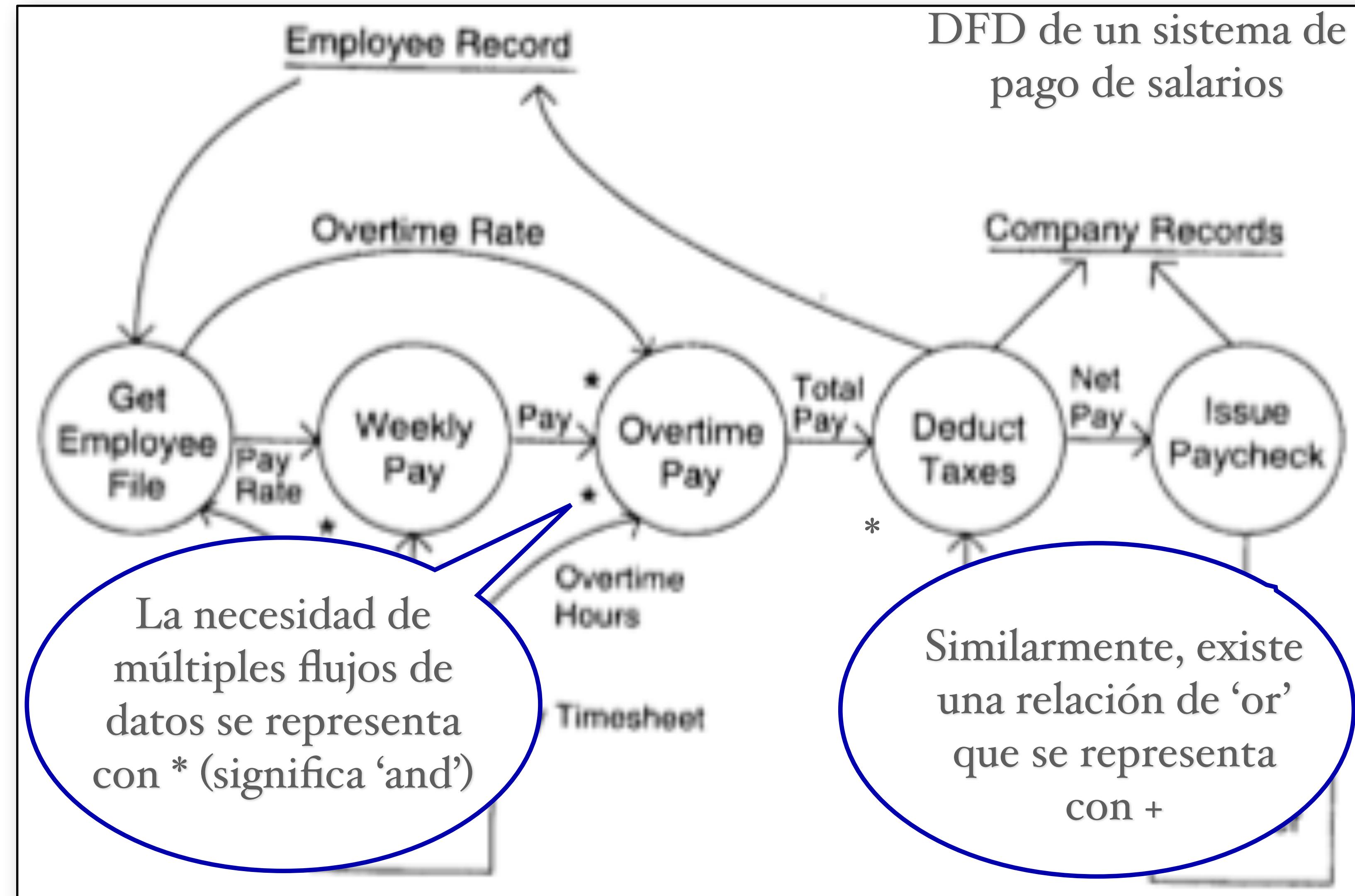
# Análisis del problema

## Modelado de flujo de datos: DFD



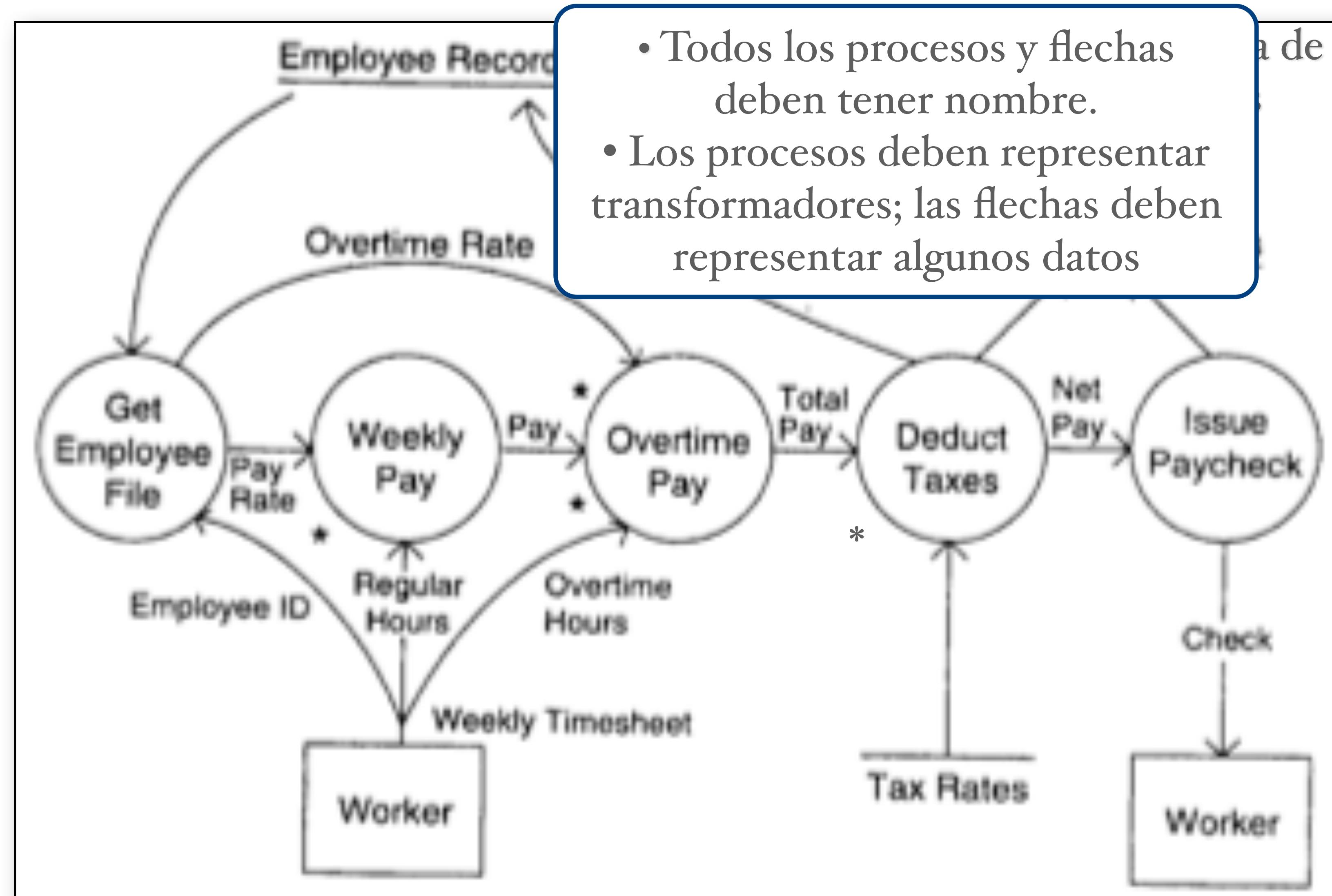
# Análisis del problema

## Modelado de flujo de datos: DFD



# Análisis del problema

## Modelado de flujo de datos: DFD



# Análisis del problema

Modelado de flujo de datos: DFD

- Se enfoca en qué hacen los transformadores, no en cómo lo hacen.
- Usualmente se muestran sólo las entradas/salidas más significativas y se ignoran las de menor importancia, como por ej. mensajes de error.
- En general, **NO** hay loops, razonamiento condicional.
- Un DFD **NO** es un diagrama de control, no debería existir diseño ni pensamiento algorítmico.

# Análisis del problema

## Modelado de flujo de datos: DFD

Cómo dibujar el DFD de un sistema:

- Identificar las entradas, salidas, fuentes, sumideros del sistema.
- Trabajar consistentemente desde la entrada hacia la salida.
- Si se complica => cambiar el sentido, de la salida a la entrada.
- Avanzar identificando los transformadores de más alto nivel para capturar la transformación completa.
- Cuando los transformadores de alto nivel están definidos, refinar cada uno con transformaciones más detalladas.
- No mostrar nunca lógica de control; si se comienza a pensar en término de loops/condiciones: parar y recomenzar.
- Etiquetar cada flecha y burbuja. Identificar cuidadosamente las entradas y salidas de cada transformador.
- Hacer uso de + y \*.
- Intentar dibujar grafos de flujo de datos alternativos antes de definirse por uno.

# Análisis del problema

## Modelado de flujo de datos: DFD

DFD en niveles:

- El DFD de un sistema puede resultar muy grande.  
=> organizar jerárquicamente.
- Comenzar con un DFD de nivel superior abstracto conteniendo pocas burbujas.
- Luego dibujar un DFD por cada burbuja.
- Al “explotar” una burbuja, preservar la E/S original con el fin de preservar consistencia.
- Para obtener el DFD en niveles se realiza un proceso de refinamiento top-down. Esto permite modelar sistemas grandes y complejos.

# Análisis del problema

Modelado de flujo de datos: DFD

El diccionario de datos:

- Las flechas del DFD están etiquetadas con ítems de datos.
- El diccionario de datos define con mayor precisión los datos en un DFD.
- Muestra la estructura de los datos; la estructura se hace más visible a medida que se van “explotando”, refinando, las burbujas.
- Se puede usar expresiones regulares para representar la estructura de datos.

# Análisis del problema

## Modelado de flujo de datos: DFD

El diccionario de datos:

- Algunos ejemplos del sistema de pago de salarios:

```
weekly_timsheet =  
    Employee_name +  
    Employee_Id +  
    [Regular_hours + Overtime_hours] *  
  
pay_rate =  
    [Hourly | daily | weekly] +  
    Dollar_amount  
  
Employee_name =  
    Last + First + Middle_initial  
  
Employee_Id =  
    digit + digit + digit + digit
```

# Análisis del problema

## Modelado de flujo de datos: DFD

Errores comunes al graficar DFD:

- Flujos de datos sin etiquetar.
- Flujos de datos omitidos, necesarios para un proceso.
- Flujos de datos irrelevantes, dibujado pero no usado por el proceso.
- Consistencia no preservada por el refinamiento.
- Procesos omitidos.
- Inclusión de información de control.

# Análisis del problema

Modelado de flujo de datos: Método de análisis estructurado

- Fue muy usado para automatizar sistemas manuales ya existentes.
- Pasos principales:
  1. Dibujar el diagrama del contexto.
  2. Dibujar el DFD del sistema existente.
  3. Dibujar el DFD del sistema propuesto e identificar la frontera hombre-máquina.

# Análisis del problema

Modelado de flujo de datos: Método de análisis estructurado

## I) Diagrama de Contexto:

- Ve al sistema completo como un transformador e identifica el contexto.
- Es un DFD con un único transformador (el sistema), con entradas, salidas, fuentes, y sumideros del sistema identificado.

# Análisis del problema

Modelado de flujo de datos: Método de análisis estructurado

## 2) DFD del sistema existente:

- El sistema actual se modela tal como es con un DFD con el fin de comprender el funcionamiento.
- Se refina el diagrama de contexto.
- Cada burbuja representa una transformación lógica de algunos datos.
- Pueden usarse DFD en niveles jerárquicos.
- Para obtenerlo se debe interactuar intensamente con el usuario.
- El DFD obtenido se valida junto a los usuarios, haciendo una “caminata” a través del DFD.

# Análisis del problema

Modelado de flujo de datos: Método de análisis estructurado

## 3) Modelado del sistema propuesto:

- No existen reglas generales para dibujar el DFD del futuro sistema  
=> hay que ser creativo.
- Utilizar conocimiento/comprepción existente.
- El DFD debe modelar el sistema propuesto completo: ya sean procesos automatizados o manuales.
- Validar con el usuario, también esta parte.
- Establecer luego la frontera hombre máquina: qué procesos se automatizarán y cuáles permanecerán manuales.
- Mostrar claramente la interacción entre los procesos manuales y los automáticos.

# Análisis del problema

## Modelado de flujo de datos: Ejemplo

La dueña de un restaurante cree que, automatizando algunas parte del negocio mejorará la eficiencia de su negocio. Ella también cree que un sistema automatizado puede hacer el negocio más atractivo para el cliente. Por lo tanto ella desea automatizar su restaurante lo más posible.

Aquí viene donde realizamos entrevistas y cuestionarios con el cliente y los usuario, y recolectamos la información en general.

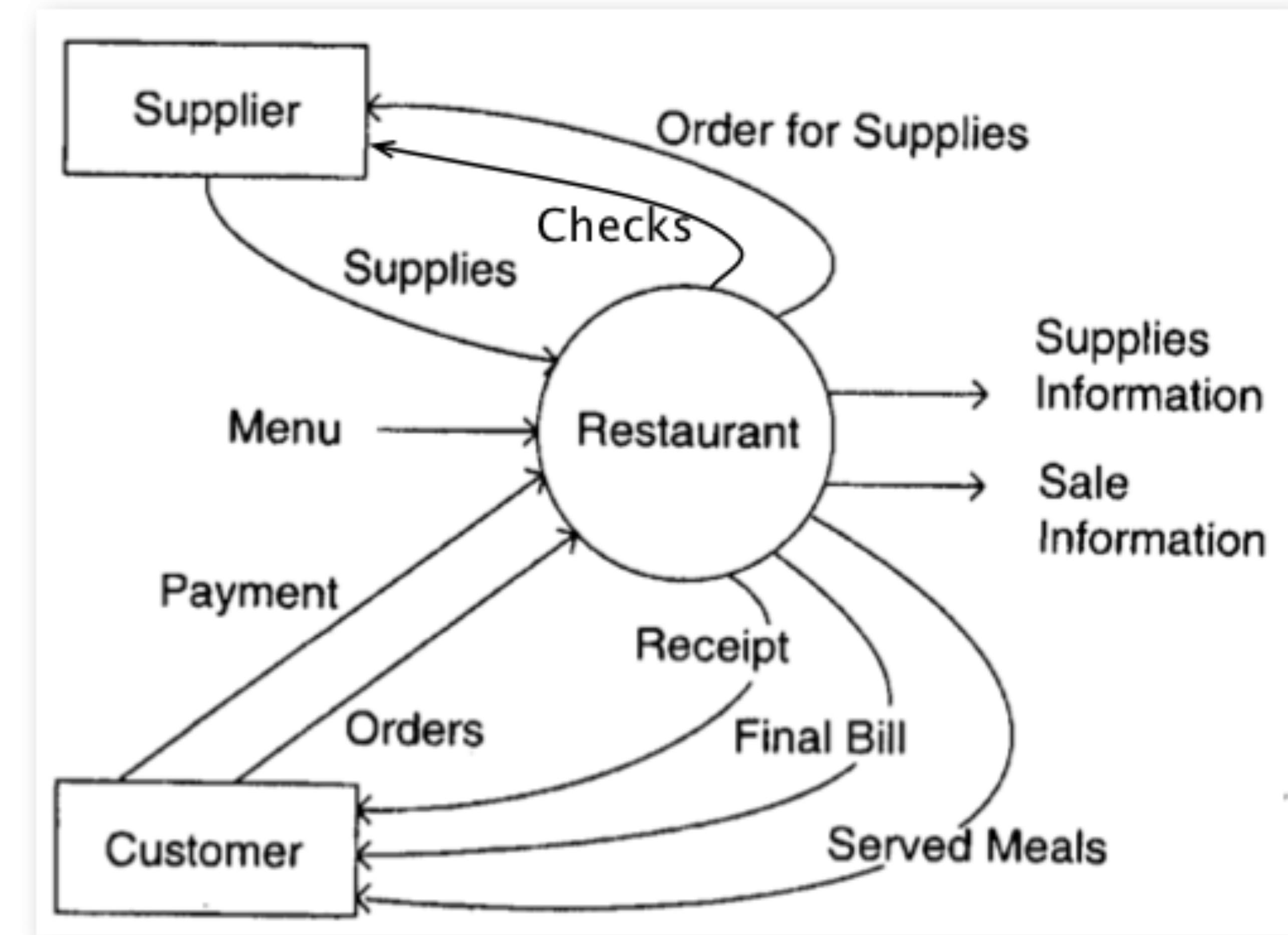
Primero identificamos las partes involucradas:

- Cliente: la dueña del restaurante.
- Usuarios potenciales: mozos, operador de la caja registradora.

# Análisis del problema

Modelado de flujo de datos: Ejemplo

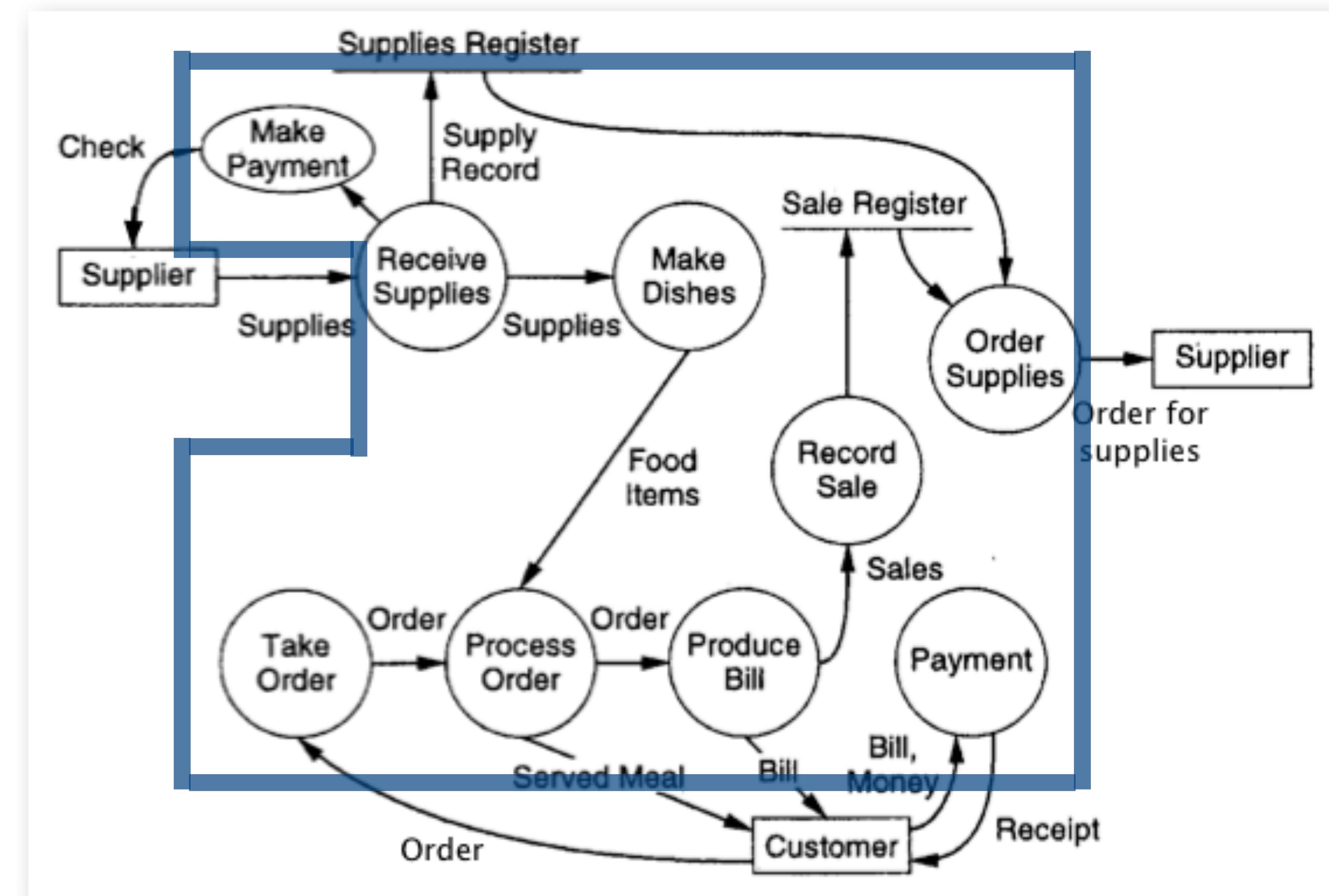
i) El diagrama de contexto:



# Análisis del problema

Modelado de flujo de datos: Ejemplo

2) DFD del sistema existente:



# Análisis del problema

## Modelado de flujo de datos: Ejemplo

### 3) DFD del sistema propuesto

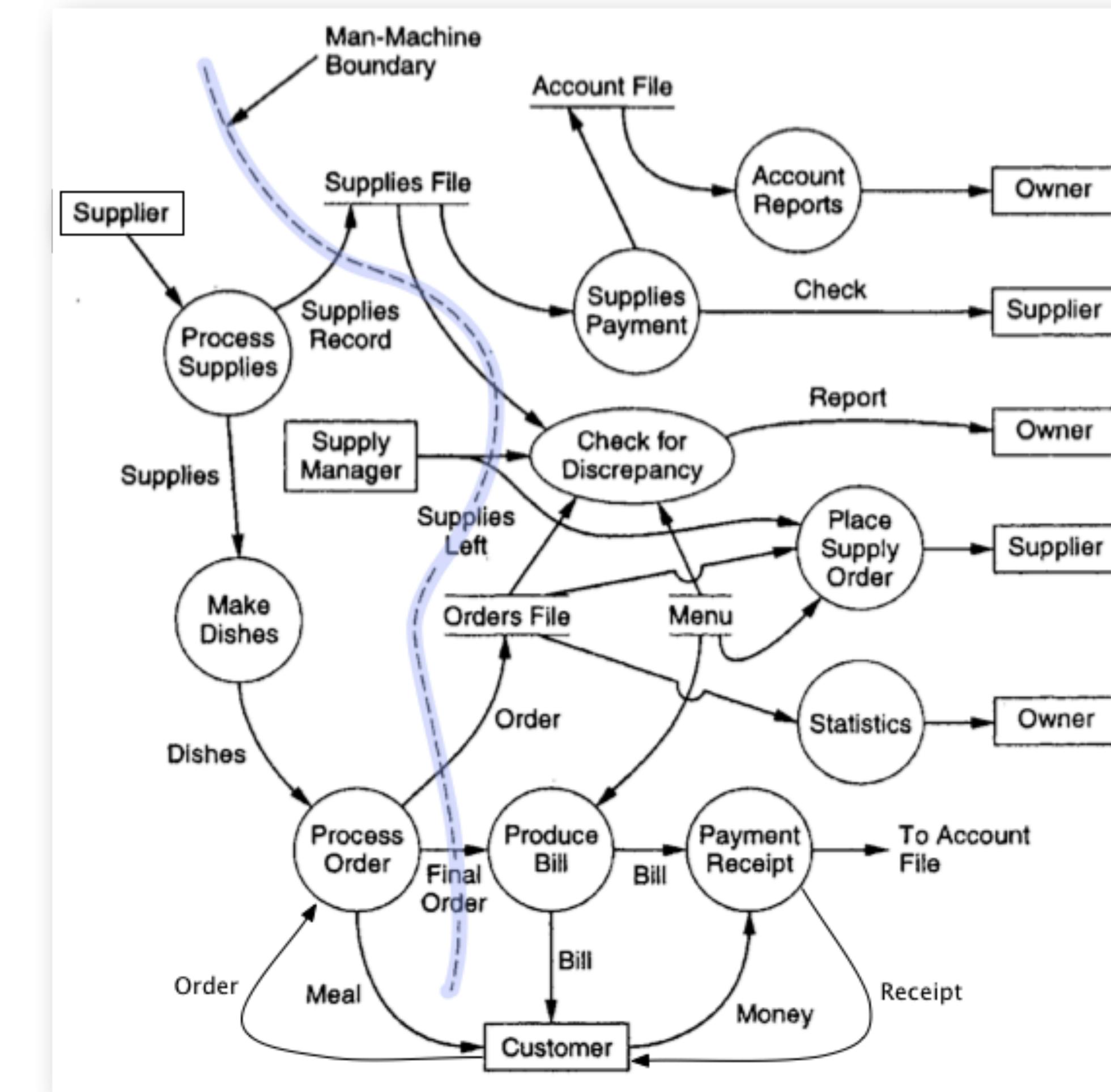
Luego de la interacción con la cliente/usuarios se concluye:

- Automatizar la mayoría del proceso de recepción y procesado de la orden y cobrado.
- Automatizar la contabilidad y stock.
- Hacer más exacto el procesamiento de suministros de manera de minimizar los desperdicios al final del día y maximizar la disponibilidad de órdenes.
- La propietaria sospecha que el personal puede estar robando suministros. Ella desea que el nuevo sistema detecte y reduzca esto.
- La propietaria también desea estadísticas sobre las ventas.

# Análisis del problema

## Modelado de flujo de datos: Ejemplo

3) DFD del sistema propuesto:



# Análisis del problema

## Modelado de flujo de datos: Ejemplo

3) Diccionario de datos del DFD del sistema propuesto:

```
Supplies_file = [date + [item_no + quantity + cost]* ]*
Orders_file = [date + [menu_item_no + quantity + status]* ]*
status = satisfied | unsatisfied
order = [menu_item_no + quantity]*
menu = [menu_item_no + name + price + supplies_used]*
supplies_used = [supply_item_no + quantity]*
bill = [name + quantity + price]* +
        total_price + sales_tax + service_charge + grand_total
discrepancy_report = [supply_item_no +
                      amt_ordered + amt_left + amt_consumed + descr]*
```

# Análisis del problema

## Modelado orientado a objetos

Ventajas del modelo orientado a objetos:

- Más fácil de construir y de mantener.
- La transición del análisis orientado a objetos al diseño orientado a objetos parece ser más simple.
- El análisis orientado a objetos es más resistente/adaptable a cambios porque los **objetos son más estables que las funciones**.

# Análisis del problema

## Modelado orientado a objetos

El sistema es visto como un conjunto de objetos interactuando entre sí, o con el usuario, a través de servicios que cada objeto provee.

Objetivo:

- Identificar los objetos, i.e. las clases, en el dominio del problema.
- Definir las clases identificando cuál es la información del estado que ésta encapsula, i.e. los atributos.
- Identificar las relaciones entre los objetos de las distintas clases, ya sea en la jerarquía o a través de llamadas a métodos.

# Análisis del problema

Modelado orientado a objetos: Conceptos básicos

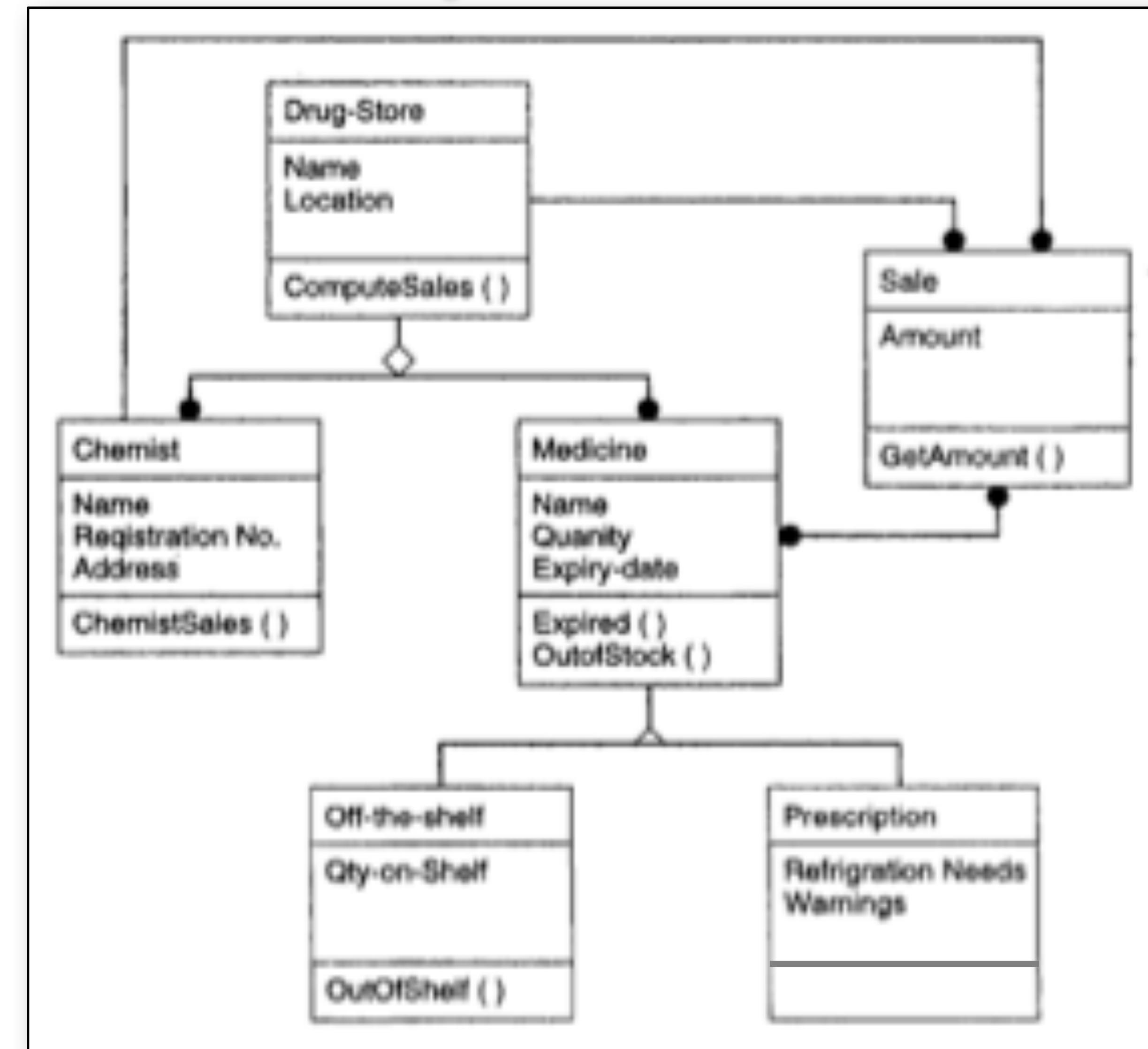
El sistema consta de objetos.

- Cada objeto tiene atributos que juntos definen al objeto. Definen el estado del objeto.
- Los objetos de tipos similares se agrupan en clases, de objetos.
- Un objeto **provee servicios o realiza operaciones**.
- Estos servicios son los únicos medios que permiten ver o modificar el estado de un objeto.
- Los servicios se acceden a través de mensajes que se envían a los objetos.

# Análisis del problema

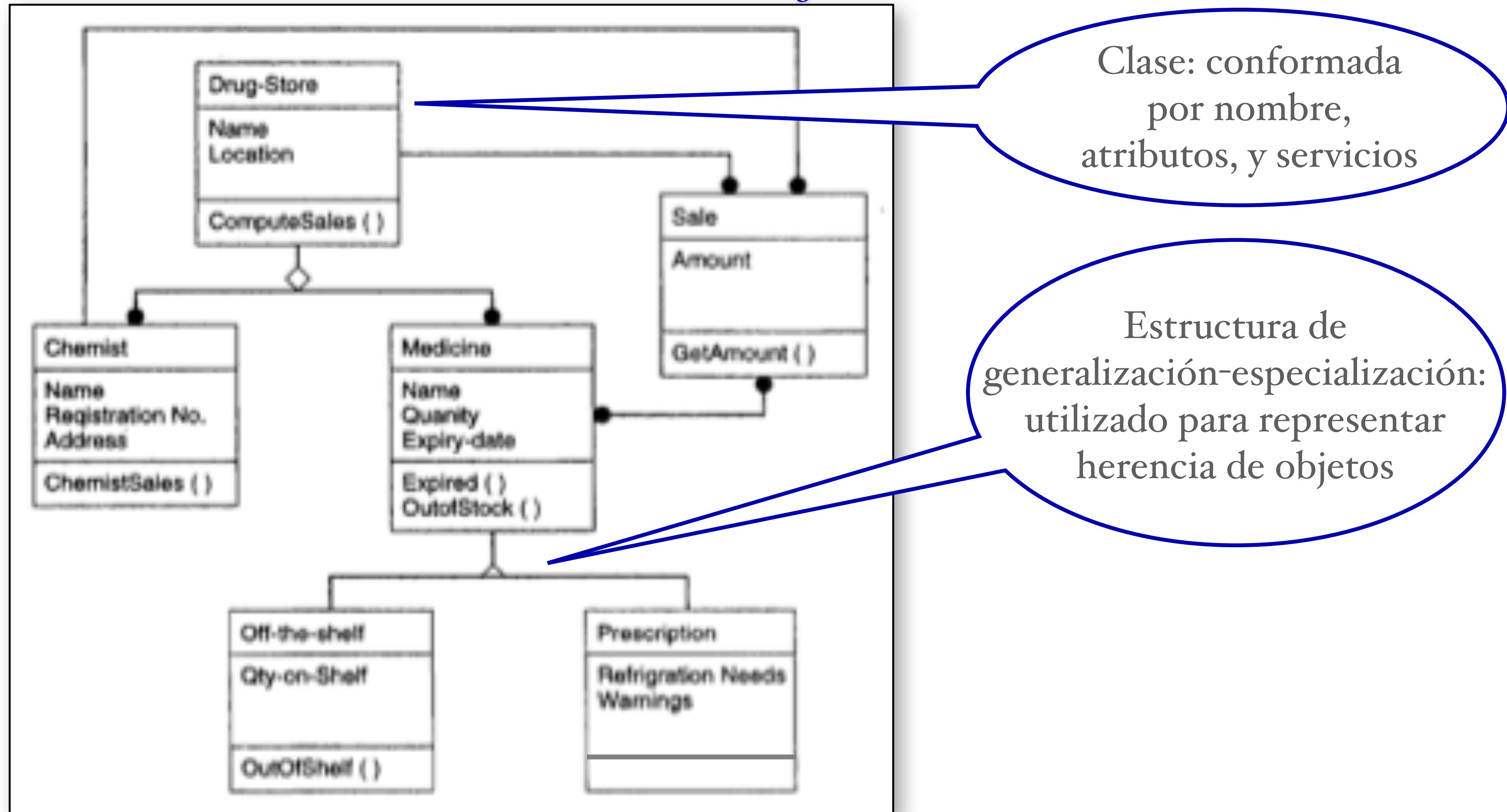
## Modelado orientado a objetos: Notación

Diagrama de clase:  
representa gráficamente  
la estructura del  
problema



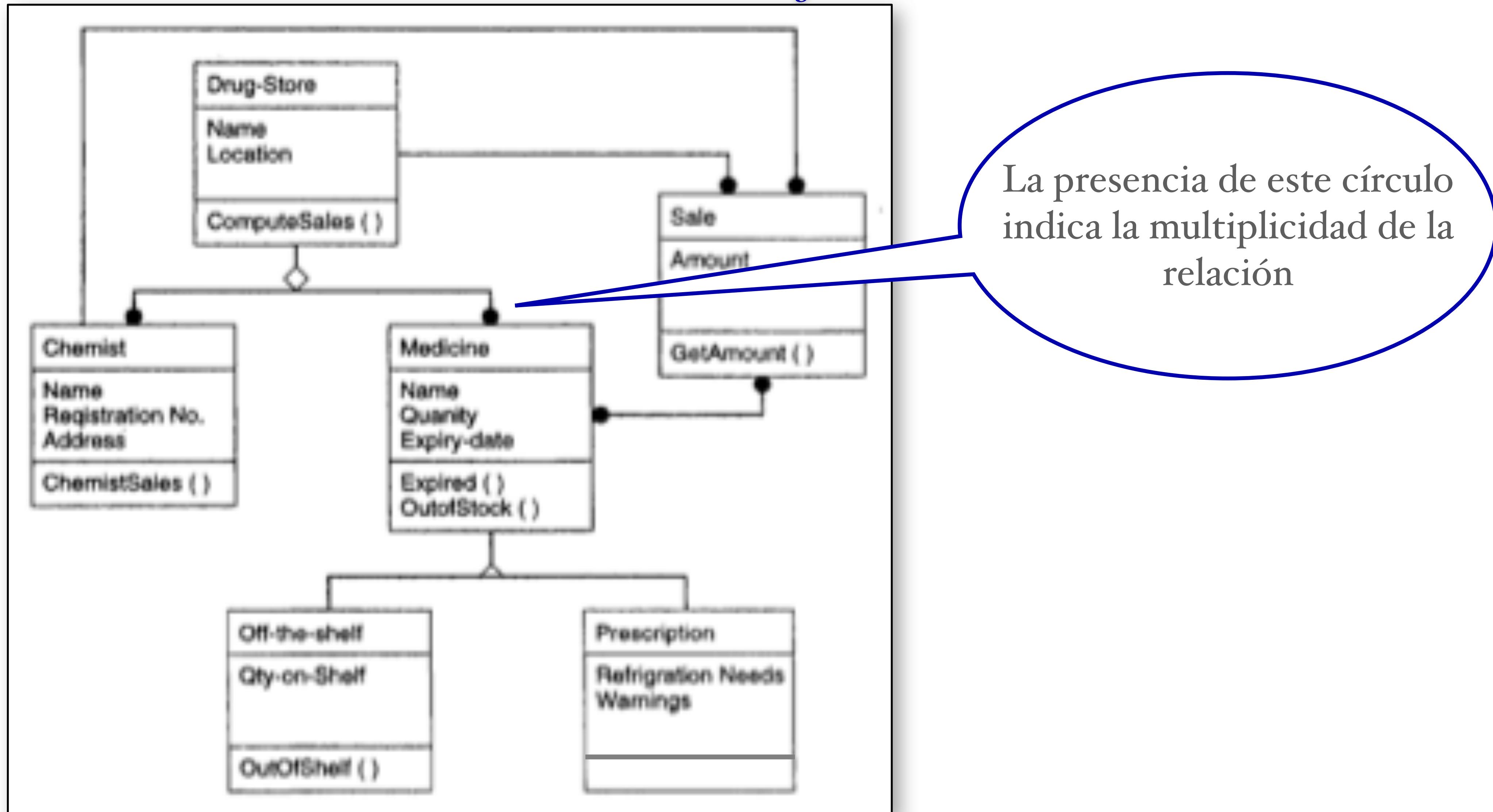
# Análisis del problema

## Modelado orientado a objetos: Notación



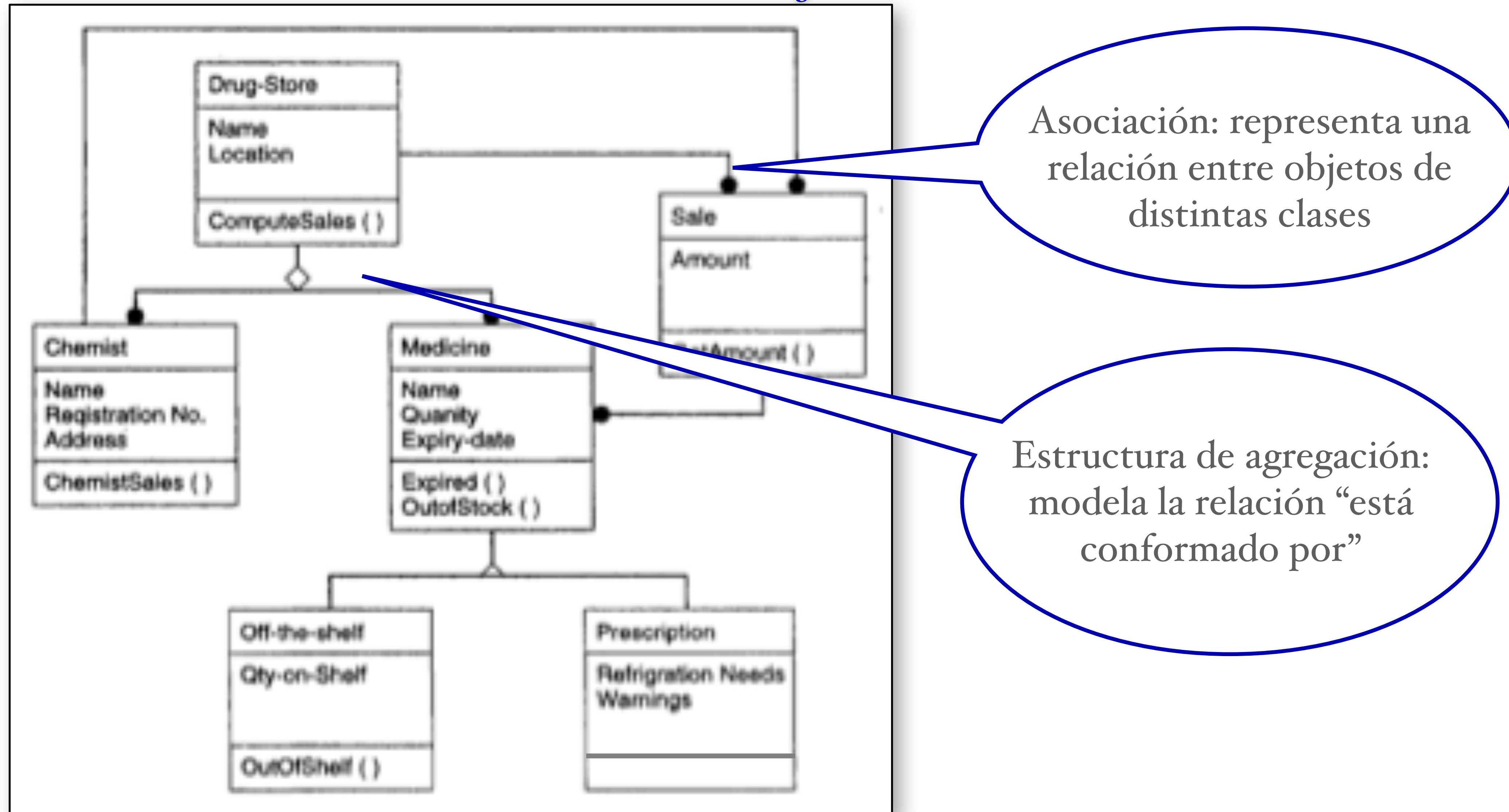
# Análisis del problema

## Modelado orientado a objetos: Notación



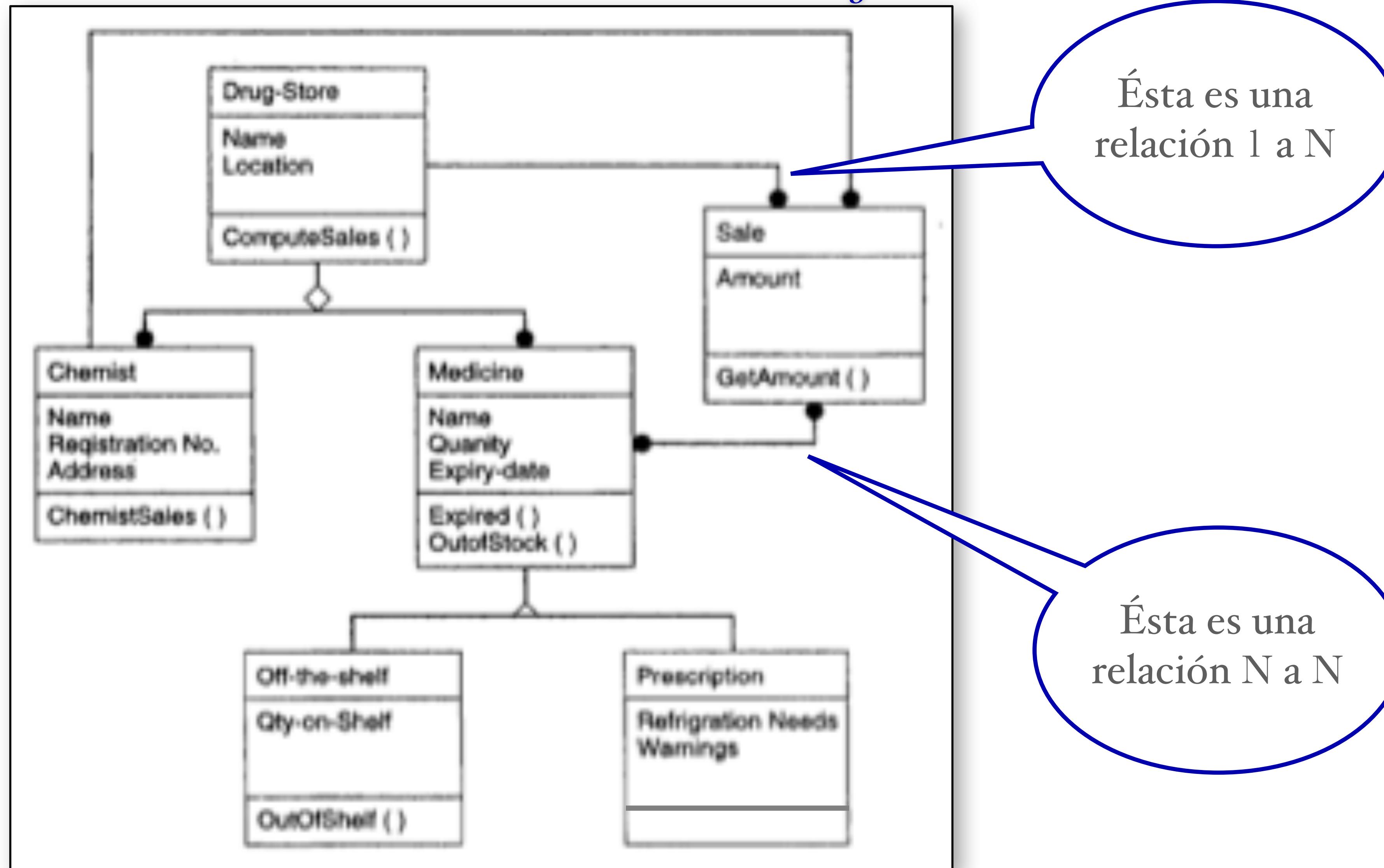
# Análisis del problema

## Modelado orientado a objetos: Notación



# Análisis del problema

## Modelado orientado a objetos: Notación



# Análisis del problema

Modelado orientado a objetos: Análisis

Los pasos más significativos para realizar el análisis orientado a objetos son:

- Identificar objetos y clases.
- Identificar estructuras.
- Identificar atributos.
- Identificar asociaciones.
- Definir servicios.

Observar el espacio del problema -  
Obtener breve resumen - Identificar  
sustantivos - Aislar candidatos como  
potenciales objetos del sistema -  
Posteriormente podrían descartarse

Considerar clases previamente  
identificadas - Considerar si alguna  
generaliza/especializa a otra (que sea  
significativa!) - Considerar si algún  
objeto es parte de otro para definir  
agregación

# Análisis del problema

## Modelado orientado a objetos: Análisis

Los pasos más significativos para realizar el análisis orientado a objetos son:

- Identificar objetos y clases.
- Identificar estructuras.
- Identificar atributos.
- Identificar asociaciones.
- Definir servicios.

Los atributos representan las características que definen los objetos - Considerarlos cuidadosamente dentro del dominio del problema  
- No agregar atributos innecesarios - Ubicarlos apropiadamente en la jerarquía de clases

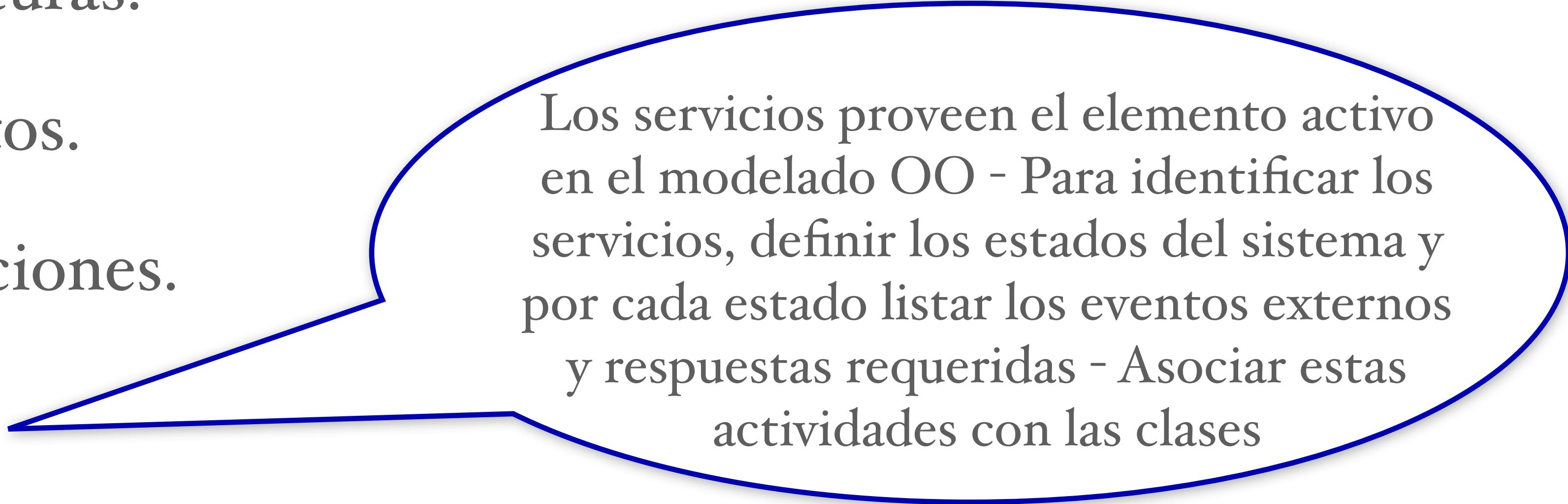
Una asociación captura la relación entre instancias de varias clases - Usualmente se derivan directamente del dominio del problema - Pueden tener sus propios atributos, no forzar estos atributos en los objetos

# Análisis del problema

Modelado orientado a objetos: Análisis

Los pasos más significativos para realizar el análisis orientado a objetos son:

- Identificar objetos y clases.
- Identificar estructuras.
- Identificar atributos.
- Identificar asociaciones.
- Definir servicios.



Los servicios proveen el elemento activo en el modelado OO - Para identificar los servicios, definir los estados del sistema y por cada estado listar los eventos externos y respuestas requeridas - Asociar estas actividades con las clases

# Análisis del problema

Modelado orientado a objetos: Ejemplo, el restaurante

Objetos potenciales:

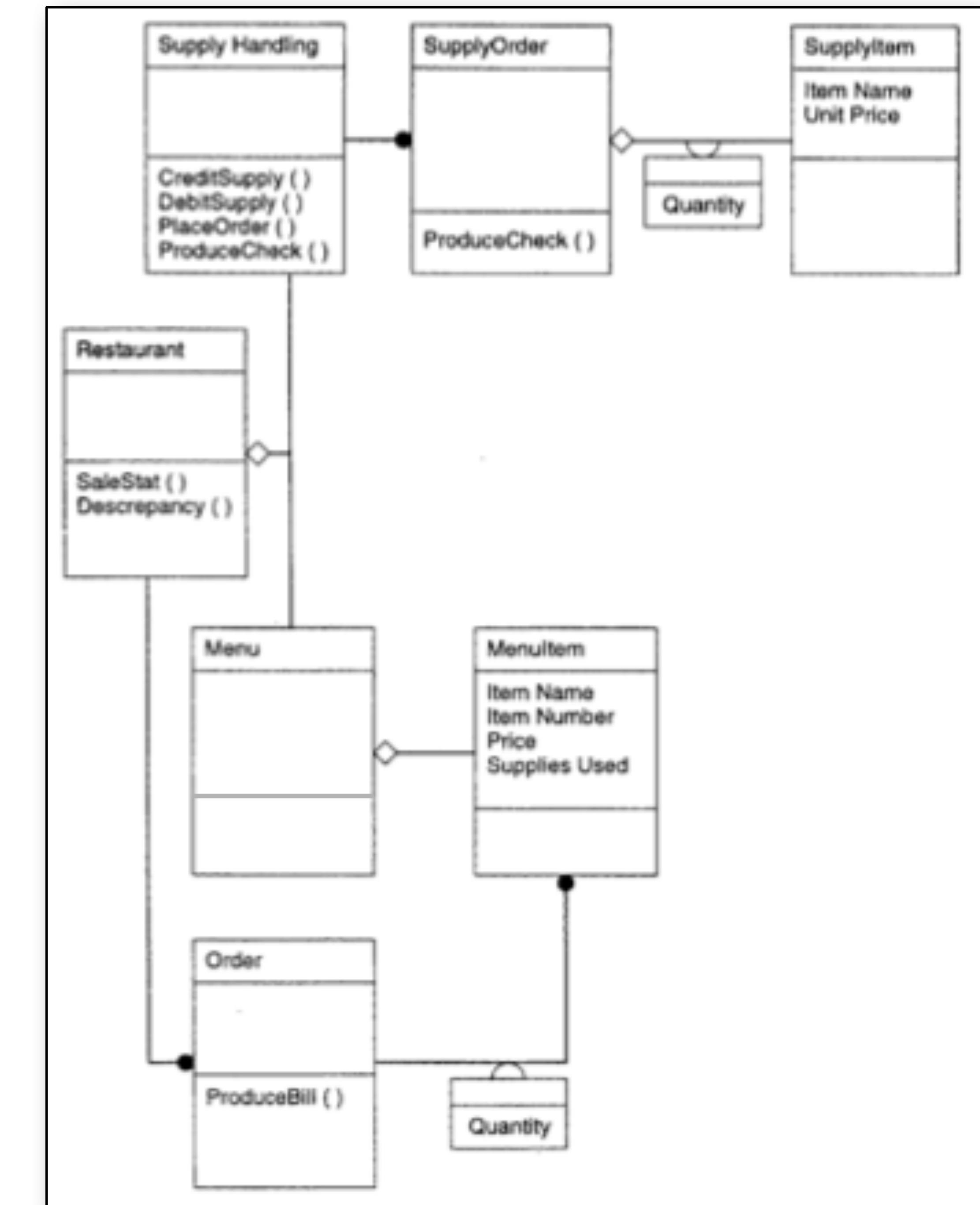
- Restaurante
- Propietario
- Factura
- Menú
- Pedido del cliente
- Provisiones
- Proveedor
- Administrador de provisiones
- Orden de provisiones
- Platos

# Análisis del problema

Modelado orientado a objetos: Ejemplo, el restaurante

Objetos potenciales:

- Restaurante
- Propietario
- ~~Factura~~
- Menú
- Pedido del cliente
- Provisiones
- ~~Proveedor~~
- Administrador de provisiones
- Orden de provisiones
- Platos



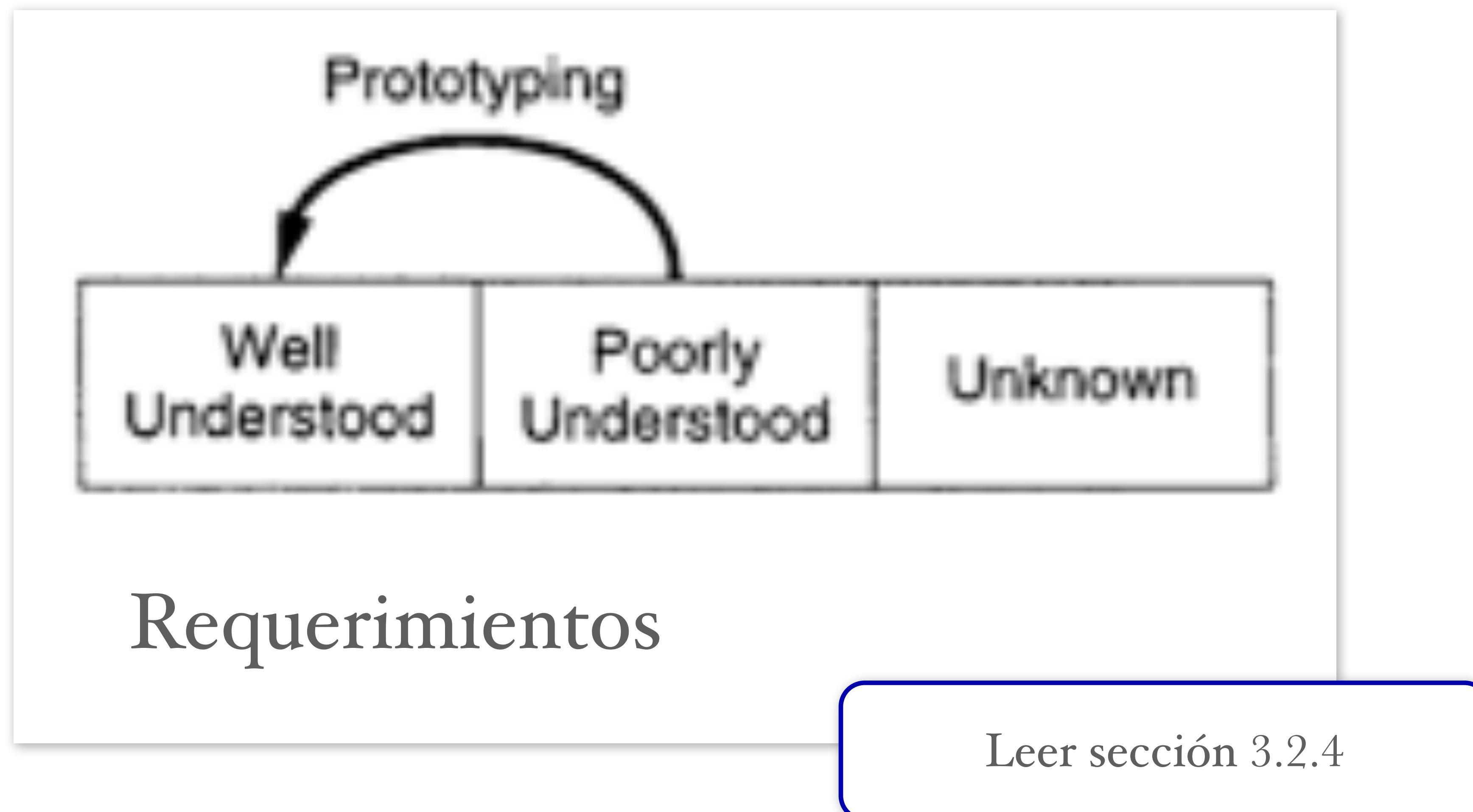
# Análisis del problema

## Prototipado

- Se construye un sistema parcial prototípico.
- Cliente, usuarios y desarrolladores lo utilizan para comprender mejor el problema y las necesidades.
- Ayuda a visualizar cómo será el sistema final.
- Dos enfoques:
  - Descartable: el prototipo se construye con la idea de desecharlo luego de culminada la fase de requerimientos.
  - Evolucionario: se construye con la idea de que evolucionará al sistema final.
- El descartable es más adecuado para esta fase del problema.

# Análisis del problema

## Prototipado



# Especificación de los requerimientos

“In principle, requirements are a relatively simple concept. They are simply statements of **what must be true** about a system to make the system acceptable...

The theory is that by defining requirements explicitly, you plan, and by planning, you save time.”

**Amy J. Ko**

# Especificación de los requerimientos

- La salida final de esta primera etapa es la SRS.
- ¿Por qué los DFD, modelos OO, o prototipado no son SRS?
  - El modelado se enfoca en la estructura del problema. La SRS se enfoca en el comportamiento externo del sistema.  
Ej.: la IU no se modela, pero debe estar en la SRS.
  - También en la SRS: tratamiento de errores, requerimientos en el desempeño, restricciones en el diseño, conformidad de estándares, recuperación, etcétera.
- La transición del modelo a la SRS no es directa.
- La SRS no es una formalización del modelo.
- Lo que se transporta del análisis a la especificación es el **conocimiento adquirido** sobre el sistema.

# Especificación de los requerimientos

## Características de una SRS

- Correcta
- Completa
- No ambigua
- Consistente
- Verificable
- Rastreable (Traceable)
- Modificable
- Ordenada en aspectos de importancia y estabilidad

# Especificación de los requerimientos

## Características de una SRS

- Corrección:
  - Cada requerimiento representa precisamente alguna característica **deseada por el cliente** en el sistema final.
- Completitud:
  - Todas las características deseadas por el cliente están descriptas.
  - La característica más difícil de lograr, para conseguirla uno debe detectar las ausencias en la especificación.
  - Corrección y completitud están fuertemente relacionadas.
- No ambigua:
  - Si para cada requerimiento existe un solo significado.
  - Si es ambigua los errores se colarán fácilmente. La no ambigüedad es esencial para verificabilidad.
  - Como la verificación es usualmente hecha a través de revisiones, la SRS debe ser comprensible, al menos por el desarrollador, el usuario y el cliente. Particular atención si se usa lenguaje natural.
  - Los lenguajes formales ayudan a “desambiguar”.

# Especificación de los requerimientos

## Características de una SRS

- Consistente:  
Ningún requerimiento contradice a otro.  
Ej.: conflictos lógicos, temporales, de dependencias.
- Verificable:  
Si existe para cada requerimiento algún proceso efectivo que puede asegurar que el software final satisface el requerimiento.
- Rastreable:  
Se debe poder determinar el origen de cada requerimiento y cómo éste se relaciona a los elementos del software.  
Hacia adelante: dado un requerimiento se debe poder detectar en qué elementos de diseño o código tiene impacto.  
Hacia atrás: dado un elemento de diseño o código se debe poder rastrear que requerimientos está atendiendo.

# Especificación de los requerimientos

## Características de una SRS

- Modificable:  
Si la estructura y estilo de la SRS es tal que permite incorporar cambios fácilmente preservando completitud y consistencia.  
La redundancia es un gran estorbo para modificabilidad, puede resultar en inconsistencia.
- Ordenada en aspectos de importancia y estabilidad:  
Los requerimientos pueden ser críticos, importantes pero no críticos, deseables pero no importantes.  
Algunos requerimientos son esenciales y difícilmente cambian con el tiempo. Otros son propensos a cambiar.  
=> Se necesita definir un orden de prioridades en la construcción para reducir riesgos debido a cambios de requerimientos.

# Especificación de los requerimientos

## Componentes de una SRS

¿Qué debe contener una SRS?

- Tener lineamientos sobre qué se debe especificar en una SRS ayudará a conseguir completitud.

Una SRS debe especificar requerimientos sobre:

- Funcionalidad.
- Desempeño (performance).
- Restricciones de diseño.
- Interfaces externas.

# Especificación de los requerimientos

## Componentes de una SRS

Requerimientos de funcionalidad:

- Es el “corazón” del documento de SRS; conforma la mayor parte de la especificación.
- Especifica toda la funcionalidad que el sistema debe proveer. Testing !!
- Especifica qué salidas se deben producir para cada entrada dada y las relaciones entre ellas.
- Describe todas las operaciones que el sistema debe realizar. Testing !!
- Describe las entradas válidas y las verificaciones de validez de la entrada y salida.
- Describe el comportamiento del sistema en caso de entradas inválidas, errores de cálculo u otras situaciones anormales, o en el caso de situaciones normales pero con imposibilidad de operar, ej. avión fully booked.

# Especificación de los requerimientos

## Componentes de una SRS

Requerimientos de desempeño:

- Todas las restricciones en el desempeño del sistema de software.
- Requerimientos **Dinámicos**, especifican restricciones sobre la ejecución:
  - Tiempo de respuesta.
  - Tiempo esperado de terminación de una operación dada.
  - Tasa de transferencia o rendimiento.
  - Cantidad de operaciones realizadas por unidad de tiempo.En general se especifican los rangos aceptables de los distintos parámetros, en casos normales y extremos.
- Requerimientos **Estáticos** o de capacidad, no imponen restricción en la ejecución:
  - Cantidad de terminales admitidas.
  - Cantidad de usuarios admitidos simultáneamente.
  - Cantidad de archivos a procesar y sus tamaños.
- Todos los requisitos se especifican en términos medibles => verificable.

# Especificación de los requerimientos

## Componentes de una SRS

Restricciones de diseño:

Existen factores en el entorno del cliente que pueden restringir las elecciones de diseño.

Por ejemplo:

- Ajustarse a estándares y compatibilidad con otros sistemas, limitaciones de hardware y otros recursos.
- Requerimientos de confiabilidad, tolerancia a falla, o respaldo, seguridad.

# Especificación de los requerimientos

## Componentes de una SRS

### Requerimientos de interfaces externas:

- Todas las interacciones del software con gente, hardware, y otros software deben especificarse claramente.
- La interfaz con el usuario debe recibir atención adecuada. Crear un manual preliminar indicando los comandos del usuario, los formatos de las pantallas, etcétera.
- Estos requerimientos también deben ser precisos para asegurar verificabilidad. Evitar cosas como “la interfaz debe ser amigable”.

# Especificación de los requerimientos

## Lenguajes de especificación

- Los lenguajes de especificación deben facilitar escribir SRS con las características deseadas, modificabilidad, no ambigüedad, etcétera.
- A la vez deben ser fáciles de aprender.
- Los lenguajes formales son precisos y carecen de ambigüedad pero no son muy fáciles de aprender.
- Usualmente se utiliza el lenguaje natural apoyado por documentos estructurados (estandarizados) para reducir imprecisiones y ambigüedades.
- La mayor ventaja del lenguaje natural es que tanto el cliente como los desarrolladores lo comprenden, pero es ambiguo...
- Las notaciones formales se utilizan en propiedades específicas del sistema o en sistemas críticos.

# Especificación de los requerimientos

## Lenguajes de especificación

- Los lenguajes de especificación deben facilitar escribir SRS con las características deseadas: modificabilidad, no ambigüedad, etcétera.
- A la vez deben ser fáciles de aprender.
- Los lenguajes formales son precisos y carecen de ambigüedad pero no son muy fáciles de aprender.
- Usualmente se utiliza el lenguaje natural apoyado por documentos estructurados (estandarizados) para reducir imprecisiones y ambigüedades.
- La mayor ventaja del lenguaje natural es que tanto el cliente como los desarrolladores lo comprenden, pero es ambiguo...
- Las notaciones formales se utilizan en propiedades específicas del sistema o en sistemas críticos.

# Especificación de los requerimientos

## Lenguajes de especificación

- Los lenguajes de especificación deben facilitar escribir SRS con las características deseadas: modificabilidad, no ambigüedad, etcétera.
- A la vez deben ser fáciles de aprender.
- Los lenguajes formales son precisos y fáciles de aprender. Ej.: expresiones regulares para especificar formatos de E/S, autómatas (o álgebras de proceso) para especificar protocolos.  
Otros: notación Z, método B, Alloy, redes de Petri, etcétera.
- Usualmente se utiliza el lenguaje natural (estandarizados) para reducir la ambigüedad. Usualmente herramientas que los soportan ofrecen lo contrario.
- La mayor ventaja del lenguaje natural es que casi todos lo comprenden, pero es ambiguo.
- Las notaciones formales se utilizan en propiedades específicas del sistema o en sistemas críticos.

# Especificación de los requerimientos

## Alcance

- Qué cosas entran en / abarca el proyecto?
- Qué cosas no entran en / no abarca el proyecto?
- Objetivos, entregables y requerimientos. Tiempos de entrega.
- Cúales son las prioridades para saber qué hacer primero.
- Criterio de aceptación (¿qué tipo de usuario lo aceptará?).
- Limitantes presupuestarios.

# Especificación de los requerimientos

## Estructura de un documento de requerimientos

- 1. Introducción
  - 1. Propósito
  - 2. Alcance
  - 3. Definiciones, acrónimos, y abreviaciones
  - 4. Referencias
  - 5. Visión general
- 2. Descripción global
  - 1. Perspectiva del producto
  - 2. Funciones del producto
  - 3. Características del usuario
  - 4. Restricciones generales
  - 5. Suposiciones y dependencias
- 3. Requerimientos específicos
  - ...

Organizar el documento en secciones y subsecciones

Estructuras estandarizadas: (1) ayuda a la comprensión por parte de otros, (2) ayuda a la completitud

La estructura general presentada responde a la guía de la IEEE para realizar SRS

# Especificación de los requerimientos

## Estructura de un documento de requerimientos

- 3. Requerimientos específicos
  - 1. Requerimientos de interfaz externa
    - 1. Interfaces del usuario
    - 2. Interfaces con el hardware
    - 3. Interfaces con el software
    - 4. Interfaces de comunicación
  - 2. Requerimientos funcionales
    - 1. Modo 1
      - 1. Requerimientos funcionales 1.1
      - \* .....
      - 3. Requerimientos funcionales 1.n
    - 2. Modo m
      - 1. Requerimientos funcionales 1.1
      - \* .....
      - 3. Requerimientos funcionales 1.n
  - 3. Requerimientos de desempeño
  - 4. Restricciones de diseño
  - 5. Atributos
  - 6. Otros requisitos

En cada req. func. se especifica la entrada requerida, salida esperada, interfaces, c/u con todos sus detalles

En general no se especifican algoritmos, solo la relación E/S (que puede estar dada como una fórmula o una ecuación)

Si se utilizan casos de uso (más adelante) los detalles de esta sección se reemplazan por las descripciones de los casos de uso, los cuales también siguen un formato estándar

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

- Enfoque tradicional: especificar cada función provista por el sistema.
- Casos de uso: captura el comportamiento del sistema como **interacción** de los usuarios con el sistema.
- Se enfoca sólo en la especificación de la funcionalidad.
- Puede utilizarse también durante análisis.
- Puede usarse para especificar comportamiento de empresas u organizaciones.  
Nuestro foco: solo software.
- Muy adecuado para sistemas interactivos.

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

Conceptos básicos:

- Un caso de uso captura un contrato entre el usuario y el comportamiento del sistema.
- La forma básica es textual. Existen notaciones gráficas (diagramas) como soporte.
- Útil en la recolección de requerimientos dado que a los usuarios les agrada, comprenden el formato, y reaccionan a éste fácilmente.

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

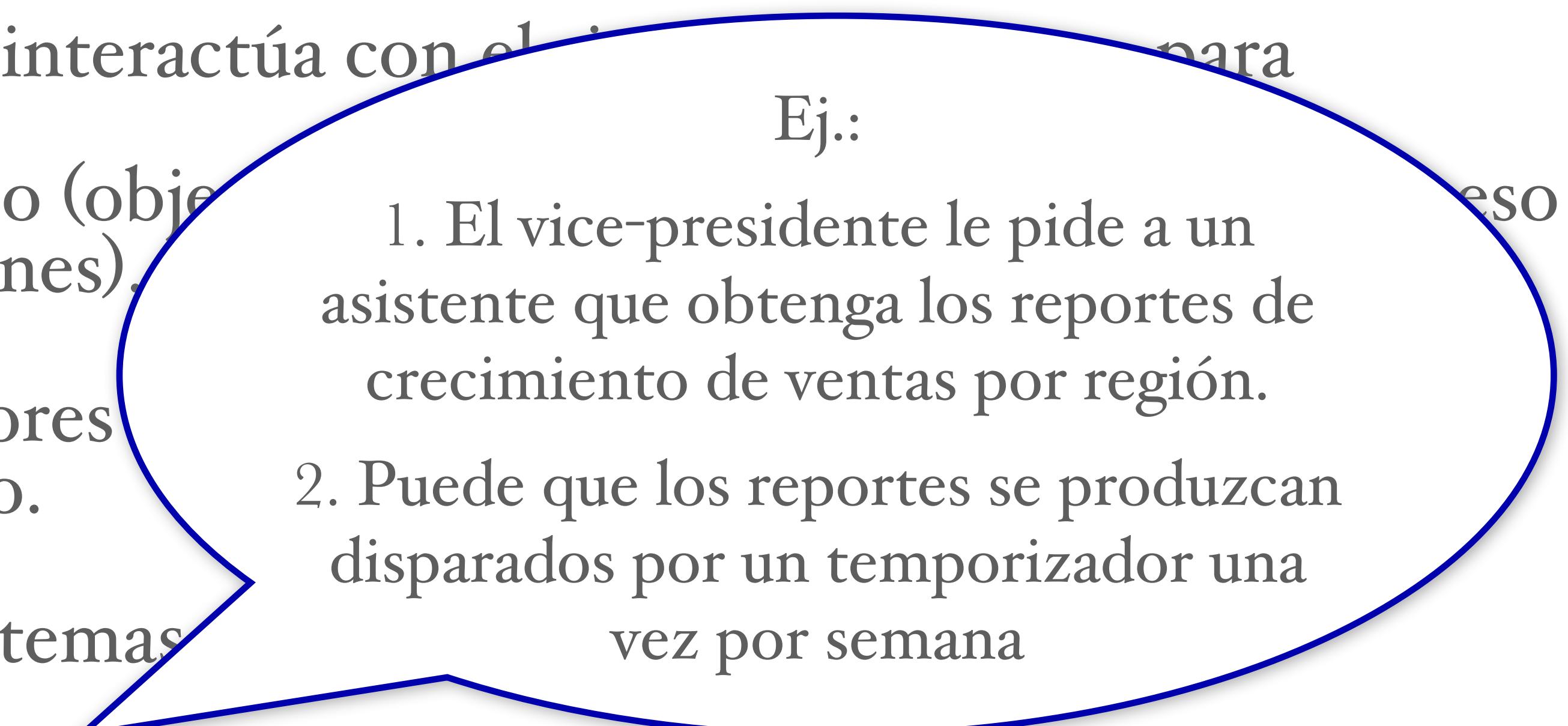
Conceptos básicos:

- **Actor:** Una persona o un sistema que interactúa con el sistema propuesto para alcanzar un objetivo.  
Ej.: El usuario de un cajero automático (objetivo: obtener dinero); operador de ingreso de datos (objetivo: realizar transacciones).
- Un actor es una entidad lógica => actores receptores y actores transmisores son distintos, aún si es el mismo individuo.
- Los actores pueden ser personas o sistemas.
- **Actor primario:** El actor principal que inicia el caso de uso.  
El caso de uso debe satisfacer su objetivo (AP es el interesado).  
La ejecución real puede ser realizada por un sistema u otra persona en representación del actor primario.

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

Conceptos básicos:

- **Actor:** Una persona o un sistema que interactúa con el sistema para alcanzar un objetivo.  
Ej.: El usuario de un cajero automático (objeto: sistema) para obtener datos (objetivo: realizar transacciones).
  - Un actor es una entidad lógica => actores distintos, aún si es el mismo individuo.
  - Los actores pueden ser personas o sistemas.
  - **Actor primario:** El actor principal que inicia el caso de uso.  
El caso de uso debe satisfacer su objetivo (AP es el interesado).  
La ejecución real puede ser realizada por un sistema u otra persona en representación del actor primario.
- 
- Ej.:
1. El vice-presidente le pide a un asistente que obtenga los reportes de crecimiento de ventas por región.
  2. Puede que los reportes se produzcan disparados por un temporizador una vez por semana

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

### Conceptos básicos:

- **Escenario:** es un conjunto de acciones realizadas con el fin de alcanzar un objetivo bajo determinadas condiciones.  
Las acciones se especifican como un conjunto de pasos.  
Un paso es una acción lógicamente completa realizada tanto por el actor como por el sistema.  
Es una interacción entre el usuario y el sistema.
- **Escenario exitoso principal:** cuando todo funciona normalmente y se alcanza el objetivo.
- **Escenarios alternativos:** (de extensión/de excepción): cuando algo sale mal y el objetivo no puede ser alcanzado.

# Especificación de los requerimientos

## Especificación funcional con escenarios

### Conceptos básicos:

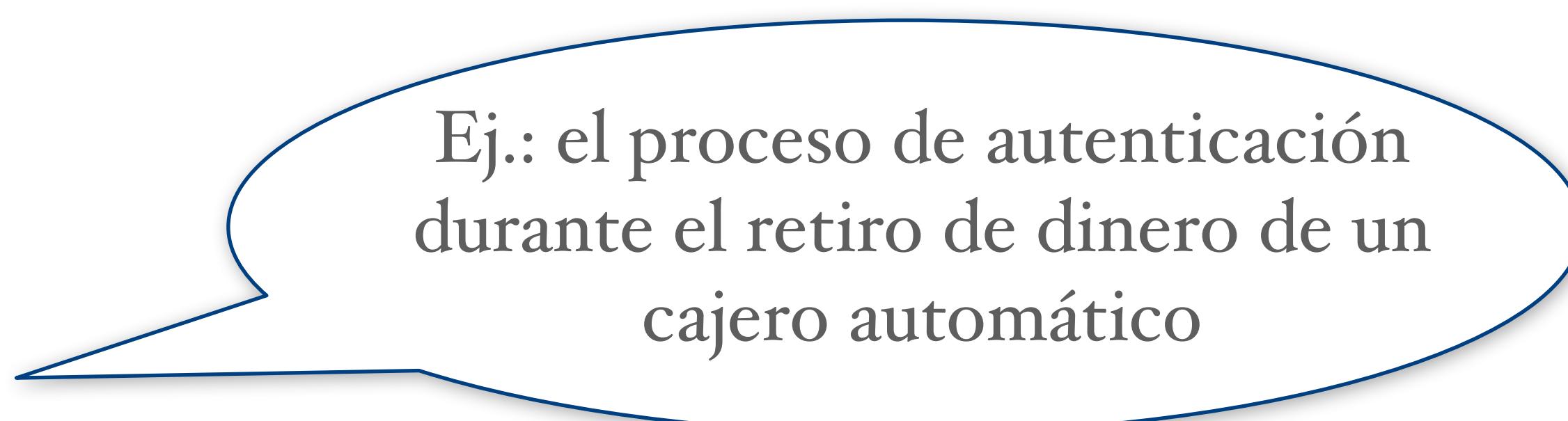
- **Escenario:** es un conjunto de acciones realizadas por un actor para alcanzar un objetivo. Generalmente se representan en secuencia, pero no necesariamente ésa es su implementación (ej.: podrían ocurrir en paralelo)
  - **Escenario exitoso principal:** cuando todo funciona como se espera.
  - **Escenarios alternativos:** (de extensión/de excepción): cuando el actor no logra alcanzar el objetivo.
- Las acciones se especifican como un **conjunto de pasos**. Un paso es una acción lógicamente completa realizada por el sistema.
- Es una interacción entre el usuario y el sistema.
- Ej.:
  1. El actor ingresar información
  2. El sistema validar información (para alcanzar objetivo) y registrar transacción (cambio de estado interno)

# Especificación de los requerimientos

## Especificación funcional con Casos de uso

Conceptos básicos:

- Un caso de uso es una colección de muchos escenarios.
- Un escenario puede emplear otros casos de usos en un paso.
- Es decir: un subobjetivo del objetivo de un caso de uso puede realizarse en otro caso de uso.
- En otras palabras: los casos de uso pueden organizarse jerárquicamente.



Ej.: el proceso de autenticación durante el retiro de dinero de un cajero automático

# Especificación de los requerimientos

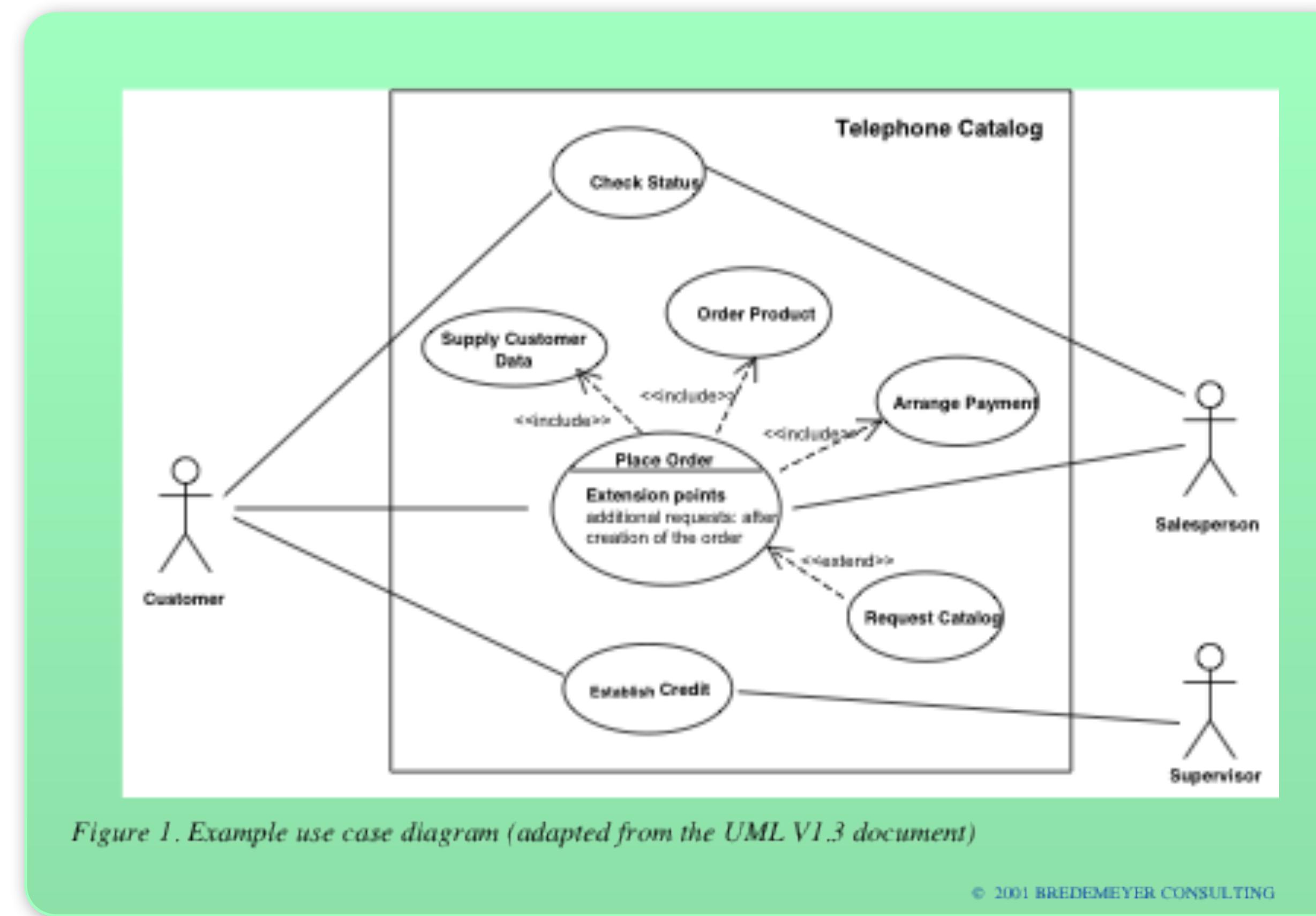
## Especificación funcional con Casos de uso

### Conceptos básicos:

- Los casos de uso especifican funcionalidades describiendo la interacción entre actores y sistema.
- Se enfocan en el comportamiento externo.
- Los casos de uso son primariamente textuales  
Los diagramas de casos de uso son suplementos de los casos de usos textuales.  
Muestran casos de usos, actores y sus dependencias.  
Sólo proveen una visión general.
- Los casos de usos **no** forman la SRS completa, sólo la parte funcional.

# Especificación de los requerimientos

## Especificación funcional con Casos de uso



# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

Pequeño sistema para subastas online:

Las personas pueden comprar y vender productos.

Suponemos un pequeño subsistema de pago donde cada comprador/vendedor tiene una cuenta.

En este sistemas tenemos “compradores” y “vendedores” como actores (y no “usuarios”), aunque sea la misma persona la que realice esas funciones.

Además, el sistema de subasta mismo es un interesado y como tal un actor.

El sistema de pago es otro actor.

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

Pequeño sistema para subastas online:

Consideraremos sólo los siguientes casos de uso:

1. Poner un ítem bajo subasta.
2. Efectuar una oferta.
3. Completar una subasta.
4. Subastar un ítem

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

Formato que usaremos para cada caso de uso

- Caso de uso #: « nombre del CU »
- Actor primario: « nombre del AP »
- Precondición: « descripción precondición »
- Escenario exitoso principal:
  - « paso 1 »
  - .....
  - « paso n »
- Escenarios excepcionales:
  - « excepción 1 »
  - .....
  - « excepción n »

Los casos de uso se numeran para referencias posteriores

El nombre del caso de uso especifica el objetivo del actor primario

El actor primario puede ser una persona o un sistema

La precondición describe lo que el sistema debe asegurar antes de iniciar el caso de uso

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

### Caso de uso 1: Poner un ítem bajo subasta

- Actor primario: Vendedor
- Precondición: El vendedor está logueado dentro del sistema
- Escenario exitoso principal:
  1. El vendedor pone un ítem en subasta (estableciendo la categoría del ítem, su descripción, foto, etc.).
  2. El sistema le muestra al vendedor viejos precios de ítems similares.
  3. El vendedor especifica el monto de oferta inicial y la fecha de cierre de la subasta.
  4. El sistema acepta el ítem y lo publica.
- Escenarios excepcionales:
  2. a) No existen ítems subastados bajo la misma categoría.
    - El sistema le informa al vendedor la situación.



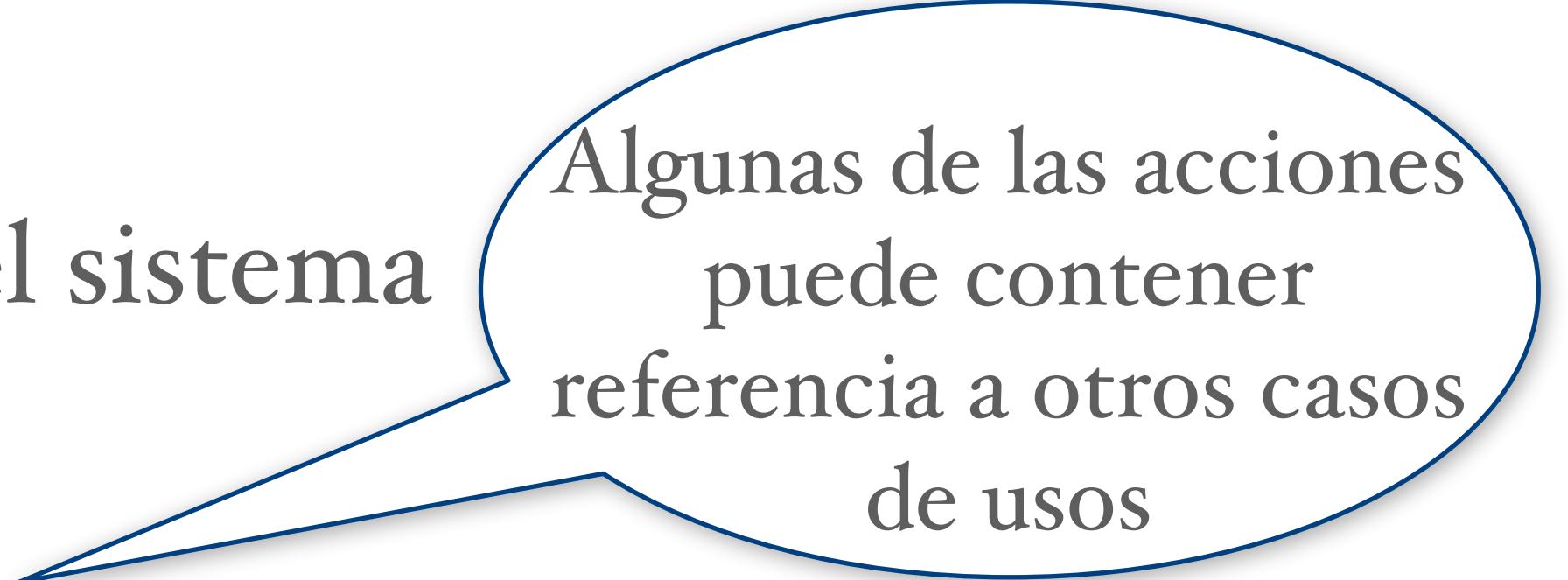
Las listas de excepciones  
no son exhaustivas

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

### Caso de uso 2: Efectuar una oferta

- Actor primario: Comprador
- Precondición: El comprador está logeado dentro del sistema
- Escenario exitoso principal:
  1. El comprador hojea o busca y elige algún ítem.
  2. El sistema le muestra la clasificación del vendedor, la oferta inicial, la oferta actual, y la oferta mayor, y le solicita al comprador que realice su oferta.
  3. El comprador especifica una oferta, el máximo monto que ofertará, y el incremento.
  4. El sistema acepta la oferta y bloquea los fondos de la cuenta del comprador, liberando los del anterior. El sistema actualiza el monto ofertado de los otros compradores cuando fuera necesario, y actualiza los registros del ítem.



Algunas de las acciones puede contener referencia a otros casos de usos

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

### Caso de uso 2: Efectuar una oferta

- Actor primario: Comprador
- Precondición: El comprador está logeado dentro del sistema.
- Escenario exitoso principal:
  1. El comprador hojea o busca y elige algún ítem.
  2. El sistema le muestra la clasificación del vendedor, la oferta y le solicita al comprador que realice su oferta.
  3. El comprador especifica una oferta, el máximo monto que ofrece.
  4. El sistema acepta la oferta y bloquea los fondos de la cuenta del comprador para el ítem anterior. El sistema actualiza el monto ofertado de los otros compradores cuando fuera necesario, y actualiza los registros del ítem.

La lista de acciones puede contener acciones que no son necesarias para el objetivo del actor primario.

Sin embargo, el sistema debe asegurar que el objetivo final, así como los otros objetivos puedan cumplirse

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

- Caso de uso 2: Efectuar una oferta

.....

3. El comprador especifica una oferta, el máximo monto que ofertará, y el incremento.
4. El sistema acepta la oferta y bloquea los fondos de la cuenta del comprador, liberando los del anterior. El sistema actualiza el monto ofertado de los otros compradores cuando fuera necesario, y actualiza los registros del ítem.

- Escenarios excepcionales:

4. a) El monto está por debajo del mayor monto ofertado.

El sistema le informa al comprador y le solicita que haga una nueva oferta.

4. b) El comprador no tiene los fondos suficientes en su cuenta.

El sistema cancela la oferta y le solicita al comprador que incremente sus fondos.

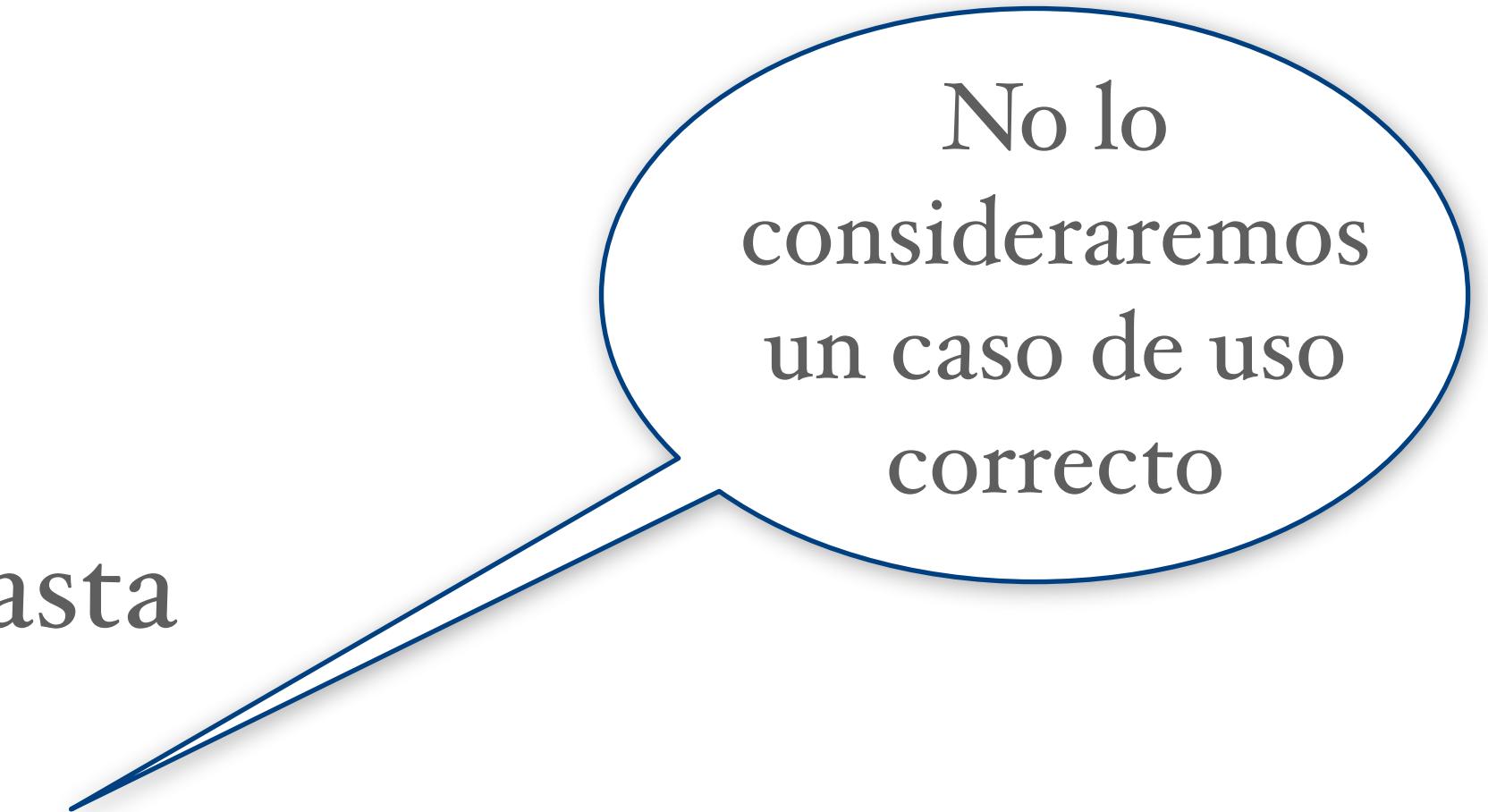
# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

### Caso de uso 3: Completar una subasta

- Actor primario: Sistema de subasta
- Precondición: Se llegó a la fecha de cierre de la subasta
- Escenario exitoso principal:
  1. Seleccionar la oferta más alta; enviar un e-mail al comprador seleccionado informándole el monto final; enviar también un e-mail a los otros compradores.
  2. Debitar de la cuenta del comprador y acreditar en la del vendedor.
  3. Transferir de la cuenta del vendedor la comisión correspondiente a la cuenta de la organización.
  4. Eliminar el ítem del sitio; actualizar registros.
- Escenarios excepcionales:

Ninguno.



No lo  
consideraremos  
un caso de uso  
correcto

# Especificación de los requerimientos

## Especificación funcional con Casos de uso - Ejemplo

Caso de uso 0: Subastar un ítem

- Actor primario: Sistema de subasta
- Ámbito: Organización conductora de la subasta
- Precondición: Ninguna
- Escenario exitoso principal:
  1. El vendedor pone un ítem bajo subasta.
  2. Varios compradores efectúan ofertas.
  3. En la fecha de cierre se completa la subasta del ítem.
  4. Obtener feedback por parte del vendedor; obtener feedback por parte del comprador; actualizar los registros.

Nivel “resumen”

El ámbito especifica el subsistema al cual se aplica el caso de uso

También es posible especificar poscondiciones

# Especificación de los requerimientos

## Elaboración de los Casos de uso

- Los casos de uso proveen un medio adecuado para la discusión y el brainstorming.  
=> también son apropiados para la recolección de requerimientos y el análisis del problema.
- Los casos de uso pueden elaborarse haciendo refinamientos paso a paso. En este contexto se presentan varios niveles de abstracción. Cuatro de ellos emergen naturalmente.

# Especificación de los requerimientos

## Elaboración de los Casos de uso - Niveles de abstracción

### 1. Actores y objetivos:

- Preparar una lista de actores y objetivos.
- Proveer un breve resumen del caso de uso.
- Esto define el ámbito del caso de uso.
- También se puede evaluar completitud.

### 2. Escenarios exitosos principales:

- Por cada caso de uso, expandir el escenario principal.
- Esto provee el comportamiento principal del sistema.
- Puede revisarse para asegurar que se satisface el interés de los participantes y actores.

# Especificación de los requerimientos

## Elaboración de los Casos de uso - Niveles de abstracción

### 3. Condiciones de falla:

- Listar las posibles condiciones de falla para cada caso de uso.
- Por cada paso, identificar cómo y por qué puede fallar.
- Este paso descubre situaciones especiales.

### 4. Manipulación de fallas: Quizás sea la parte más difícil.

- Especificar el comportamiento del sistema para cada condición de falla.
- Al realizar esta etapa pueden emergir nuevas situaciones y actores.

# Especificación de los requerimientos

## Elaboración de los Casos de uso

- Los cuatro niveles pueden dirigir el proceso de análisis comenzando desde lo más abstracto y agregando más detalles a medida que se avanza.
- Los casos de uso se deben especificar al nivel de detalle que sea suficiente, no hay un criterio general para determinar cuál es el adecuado.
- Para escribir, utilizar reglas de buena escritura técnica:  
Usar gramática simple / oraciones simples.  
Especificar claramente todas las partes del caso de uso.  
Cuando sea necesario, combinar o dividir pasos.

# CU Errores comunes

- Debe siempre haber **interacción** entre los escenarios exitosos.
- Un caso de uso nunca puede ser **iniciado** por el sistema.
- Los **escenarios excepcionales** deben siempre referenciar a un punto de los exitosos donde del sistema chequea algo.
- Las **precondiciones** de casos generales deben implicar las de los casos que se referencian dentro.
- Seguir la **sintaxis** dada en clases.
- No es **programar**, no debe haber "si pasa esto bla bla, si no pasa blabla".

# Validación de los requerimientos

- Debido a la naturaleza de esta etapa, hay muchas posibilidades de malentendidos.  
=> muchos errores son posibles.
- Es caro corregir los defectos de requerimientos más tarde.
- Se deben intentar corregir en esta etapa.
- Errores más comunes:

Omisión	30%
Inconsistencia	10-30%
Hechos incorrectos	10-30%
Ambigüedad	5-20%

A parte de los errores de “papeleo”

# Validación de los requerimientos

- La SRS se revisa por un grupo de personas.
- Grupo conformado por: autor, cliente, representantes de usuarios y de desarrolladores.
- Debe incluir al **cliente** y a los **usuarios**.
- Proceso: un proceso de inspección estándar, se verá luego.
- Efectividad: se pueden detectar entre el 40% y el 80% de los errores de requerimientos.
- Las listas de controles son muy útiles para ello.

# Validación de los requerimientos

Ejemplo de lista de control:

- ¿Se definieron todos los recursos de hardware?
- ¿Se especificaron los tiempos de respuestas de las funciones?
- ¿Se definió todo el hardware, el software externo y las interfaces de datos?
- ¿Se especificaron todas las funciones requeridas por el cliente?
- ¿Son testeables todos los requerimientos?
- ¿Se definió el estado inicial del sistema?
- ¿Se especificaron todas las respuestas a las condiciones excepcionales?
- ¿Los requerimientos contienen restricciones que pueda controlar el diseñador?
- ¿Se especifican modificaciones futuras posibles?

# Validación de los requerimientos

- Además, existen herramientas para el modelado y análisis de especificaciones.
- Se escriben en lenguajes de especificación formal.  
Ej.: Z, B, Alloy, Autómatas, Redes de Petri, Álgebras de Procesos, etcétera.
- Hay herramientas automáticas o semiautomáticas que soportan estos lenguajes.
- Permiten verificar consistencia, dependencias circulares, o propiedades específicas.
- También permiten simular para poder comprender completitud y corrección.

# Métricas

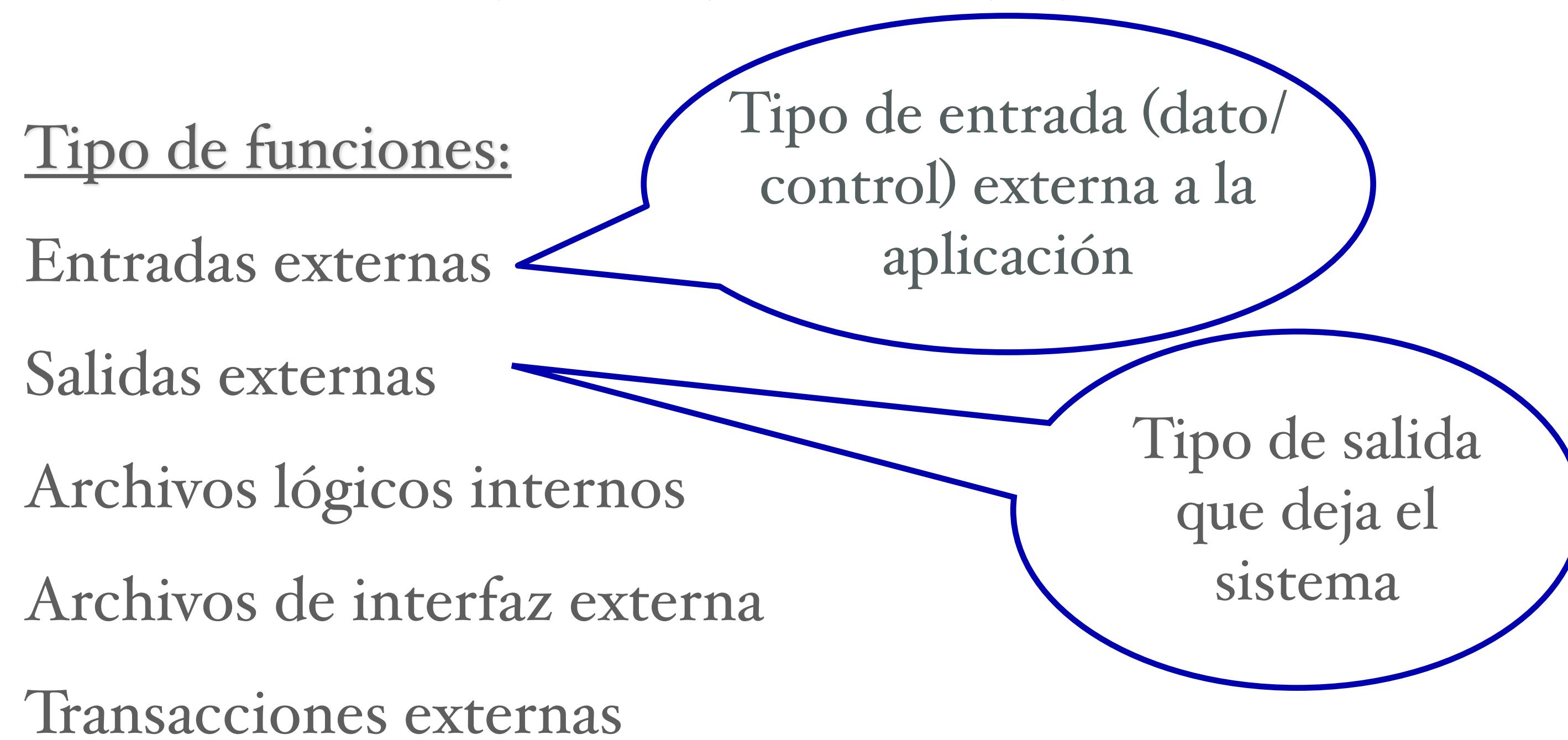
Para poder estimar costos y tiempos y planear el proyecto se necesita “medir” el esfuerzo que demandará.

- El esfuerzo del proyecto depende de muchos factores.
- El **tamaño** es el principal factor, validado por muchos experimentos y datos de análisis.
- Al principio el tamaño sólo puede ser estimado.
- Conseguir buenos estimadores es muy difícil.
- Una métrica es importante sólo si es útil para el seguimiento o control de costos, calendario o calidad.
- Se necesita una unidad de tamaño que se pueda computar a partir de los requerimientos:  
¿Tamaño de la SRS? => dependen mucho del autor

# Métricas

## Punto función

- Es una estimación similar a la métrica LOC.
- Se determina sólo con la SRS.
- Define el tamaño en términos de la “funcionalidad”.



# Métricas

## Punto función

- Es una estimación similar a la métrica LOC.
- Se determina sólo con la SRS.
- Define el tamaño en términos de la “funcionalidad”.

### Tipo de funciones:

Entradas externas

Salidas externas

Archivos lógicos internos

Archivos de interfaz externa

Transacciones externas

Grupo lógico de dato/  
control de información  
generado/usedo/  
manipulado

Archivos pasados/  
compartidos entre  
aplicaciones

Input/output  
inmediatos  
(queries)

# Métricas

## Punto función

- Es una estimación similar a la métrica LOC.
- Se determina sólo con la SRS.
- Estima el tamaño en términos de la “funcionalidad”.

<u>Tipo de funciones:</u>	Simp.	Prom.	Comp.
Entradas externas	3	4	6
Salidas externas	4	5	7
Archivos lógicos internos	7	10	15
Archivos de interfaz externa	5	7	10
Transacciones externas	3	4	6

Pesos ( $w_{ij}$ )

# Métricas

## Punto función

- Contar cada tipo de función diferenciando según sea compleja, promedio o simple.
- $C_{ij}$  denota la cantidad de funciones tipo “i” con complejidad “j”.
- Punto función no ajustado (UFP):

$$\sum_{i=1}^5 \sum_{j=1}^3 w_{ij} C_{ij}$$

# Métricas

## Punto función

- Ajustar el UFP de acuerdo a la complejidad del entorno.  
Se evalúa según las siguientes características:

1. comunicación de datos
2. procesamiento distribuido
3. objetivos de desempeño
4. carga en la configuración de operación
5. tasa de transacción
6. ingreso de datos online
7. eficiencia del usuario final
8. actualización online
9. complejidad del procesamiento lógico
10. reusabilidad
- II. facilidad para la instalación
12. facilidad para la operación
13. múltiples sitios
14. intención de facilitar cambios

- Factor de ajuste de complejidad (CAF):
- Puntos función = CAF \* UFP

1 punto función =  
• 125 LOC en C  
• 50 LOC en C++ o Java

Cada uno de estos ítems debe evaluarse como:

no presente

influencia insignificante

influencia moderada

influencia promedio

influencia significativa

influencia fuerte

$$0.65 + 0.01 \sum_{i=1}^{14} p_i$$

$p_i$

0

1

2

3

4

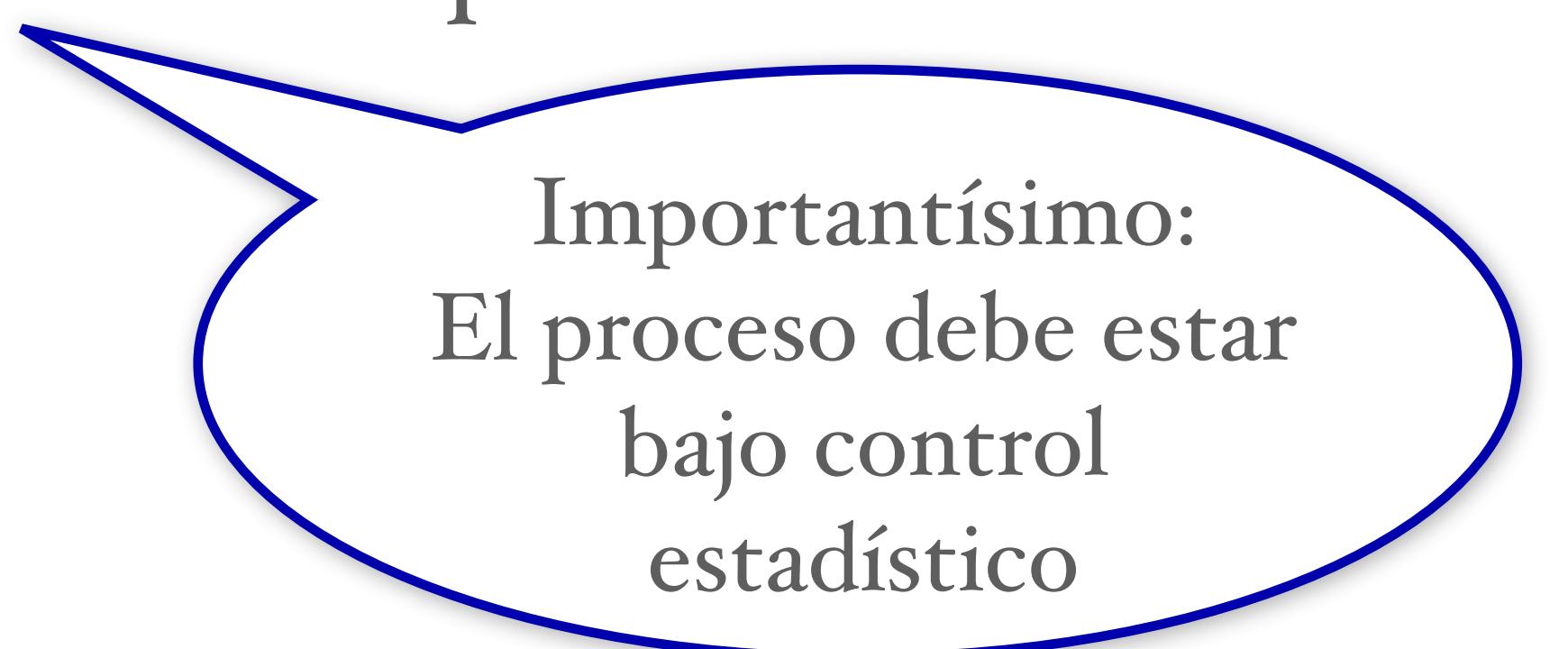
5

# Métricas

## Métricas de calidad

La calidad de la SRS tiene impacto directo en los costos del proyecto.  
=> se necesitan buenas métricas de calidad para evaluar la calidad de la SRS.

- Métricas de **calidad directa**: evalúan la calidad del documento estimando el valor de los atributos de calidad de la SRS.
- Métricas de **calidad indirecta**: evalúan la efectividad de las métricas del control de calidad usadas en el proceso en la fase de requerimientos.  
Ej.: Número de errores encontrados.  
Frecuencia de cambios de requerimientos.



Importantísimo:  
El proceso debe estar  
bajo control  
estadístico

# Material complementario

- Capítulo 3 – Jalote

- DFD

<https://juanfidel.wordpress.com/wp-content/uploads/2011/12/cocina-disec3biada-por-un-ingeniero.pdf>

- SRS

<https://www.youtube.com/watch?v=BKorP55Aqvg>

[https://www.youtube.com/watch?v=mokllJ\\_Sz\\_g&list=RDBKorP55Aqvg&index=3](https://www.youtube.com/watch?v=mokllJ_Sz_g&list=RDBKorP55Aqvg&index=3)

- Requirements by Amy J. Ko

<http://faculty.washington.edu/ajko/books/cooperative-software-development/requirements.html>