

Análisis y Diseño Orientado a Objetos

3 - Diseño

El proceso unificado de desarrollo, Ivar Jacobson, Grady Booch, James Rumbaugh, Ed. Addison Wesley, 1999

The unified software development process, Ivar Jacobson, Grady Booch, James Rumbaugh, Ed. Addison Wesley, 1999



Diseño

1. Visión general

2. El diseño en el Proceso Unificado de Desarrollo

3.1 Artefactos.

3.1.1 Modelo de diseño.

3.1.2 Clases de diseño.

3.1.3 Realización en diseño de los casos de uso.

3.1.4 Subsistemas en diseño.

3.1.5 Interfaz.

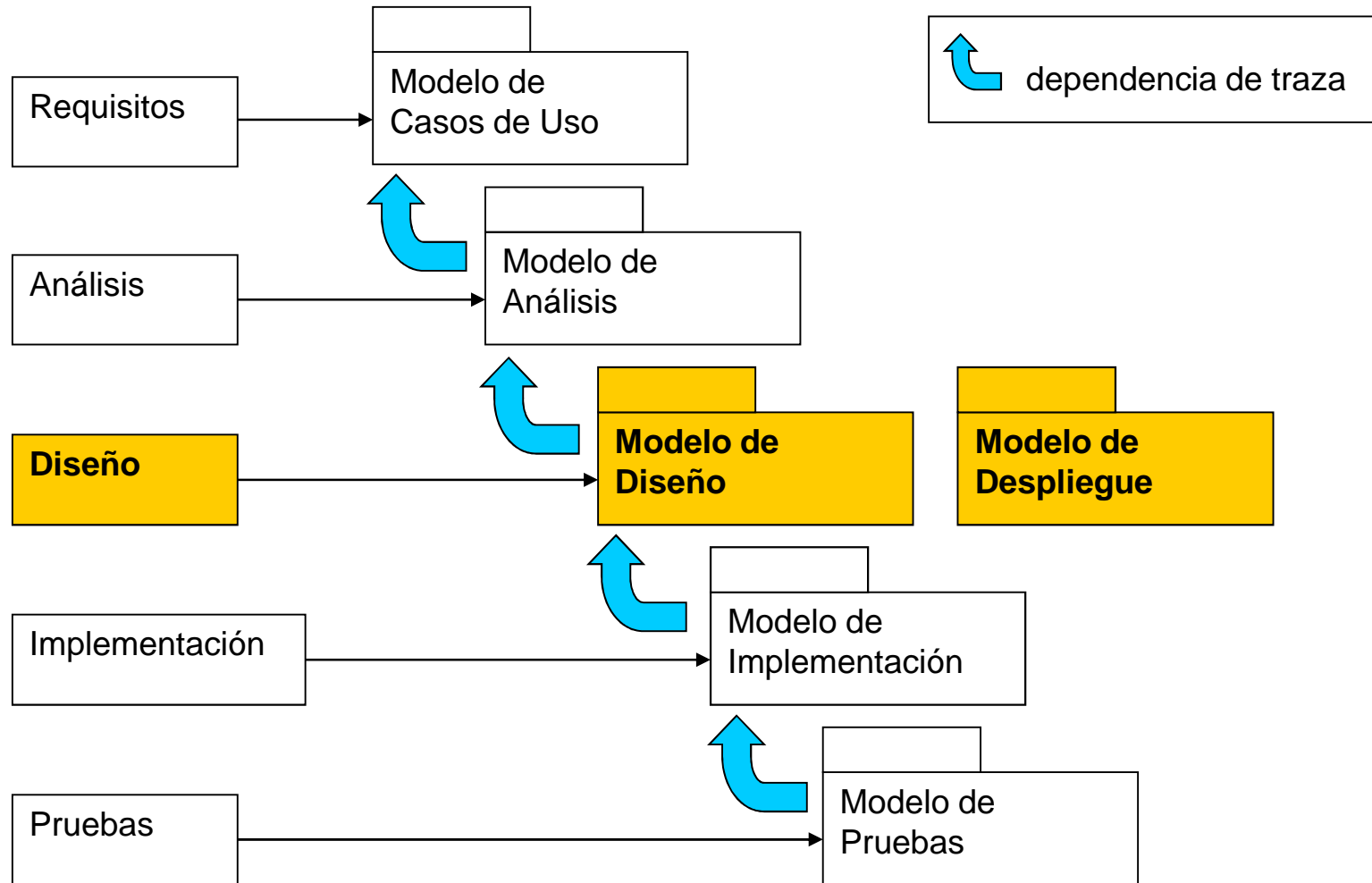
3.2 Actividades.

3.2.1. Diseño de los casos de uso.

3.2.2. Diseño de las clases.

3.2.3. Diseño de subsistemas.

1. Visión general

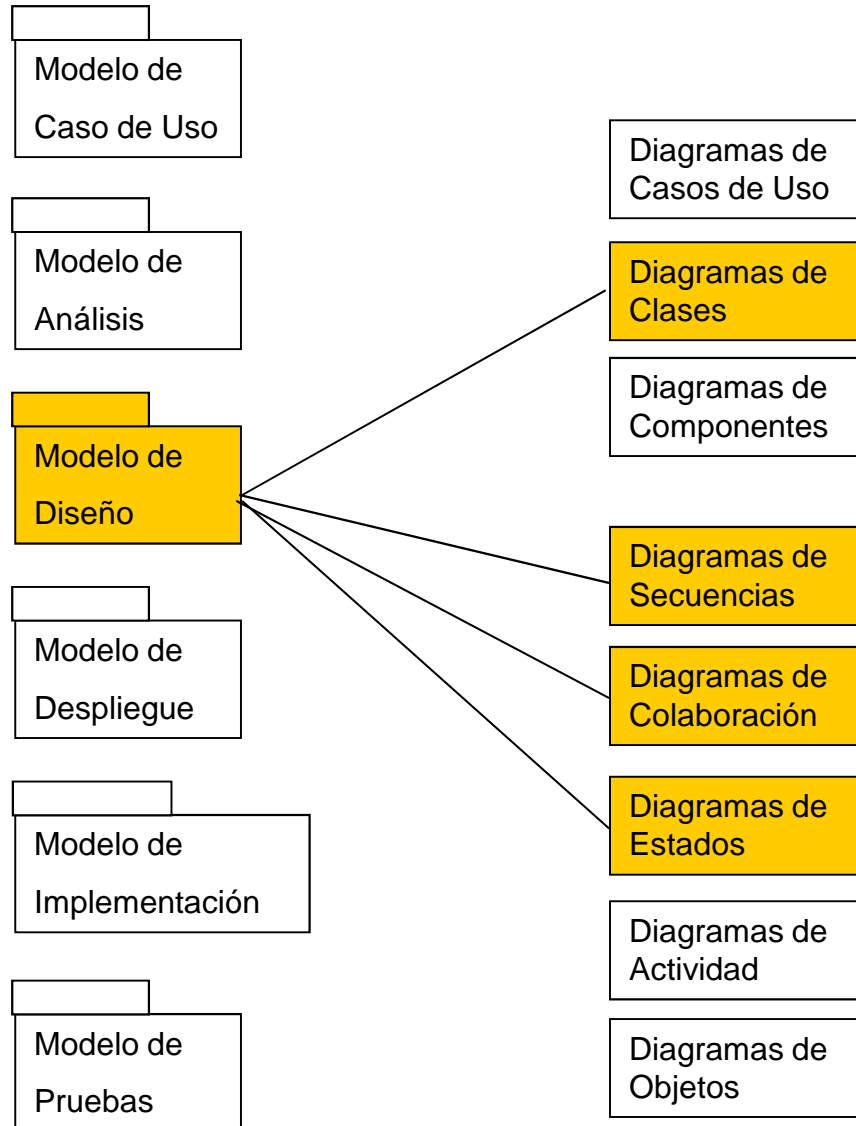




1. Visión general. Objetivos del diseño

- Acercar el modelo de análisis al modelo de implementación
 - “Los milagros más comunes de la ingeniería del software son las transiciones desde el análisis hasta el diseño y desde el diseño al código” (Richard Due).
- Identificar requisitos no funcionales y restricciones en relación a:
 - lenguajes de programación, reutilización de componentes, sistemas operativos, tecnologías de: distribución, concurrencia, bases de datos, interfaces de usuario, gestión de transacciones, etc.
- Descomponer el modelo de análisis en subsistemas que puedan desarrollarse en paralelo. Definir la interfaz de cada subsistema.
- Derivar una representación arquitectónica del sistema

Diagramas UML





Diseño

1. Visión general

2. El diseño en el Proceso Unificado de Desarrollo

3.1 Artefactos.

3.1.1 Modelo de diseño.

3.1.2 Clases de diseño.

3.1.3 Realización en diseño de los casos de uso.

3.1.4 Subsistemas en diseño.

3.1.5 Interfaz.

3.2 Actividades.

3.2.1. Diseño de los casos de uso.

3.2.2. Diseño de las clases.

3.2.3. Diseño de subsistemas.

3.1.1 Artefactos. Modelo de diseño

- **Artefactos**

- **Modelo de diseño**

- Clases de diseño

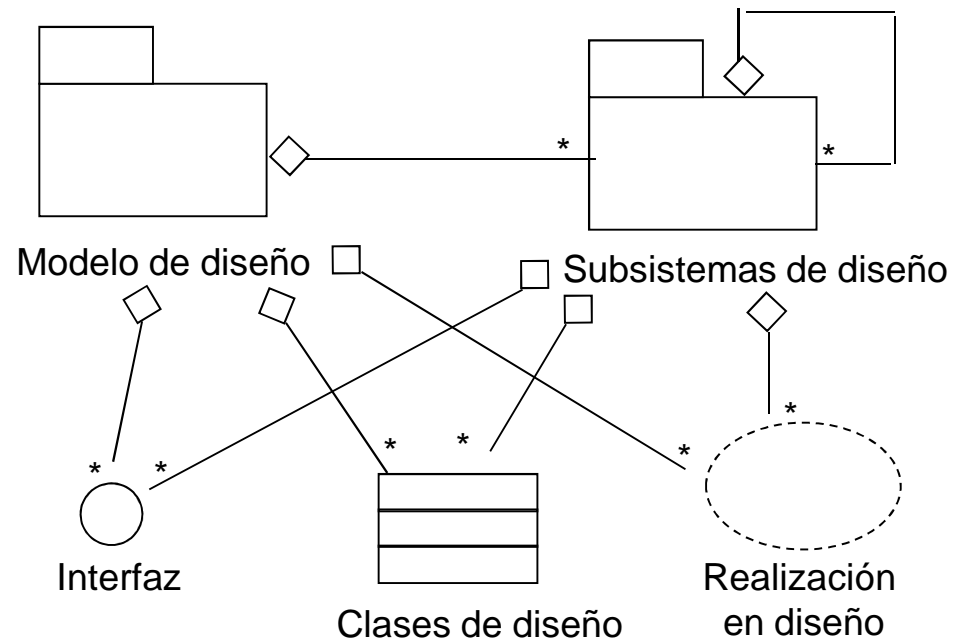
- Realización en diseño

- Subsistemas

- Interfaz

- **Actividades**

- Casos de uso en el dominio de la solución
- Cómo soportar requisitos funcionales/no funcionales y otras restricciones en el entorno de implementación
- Entrada fundamental para actividades de implementación

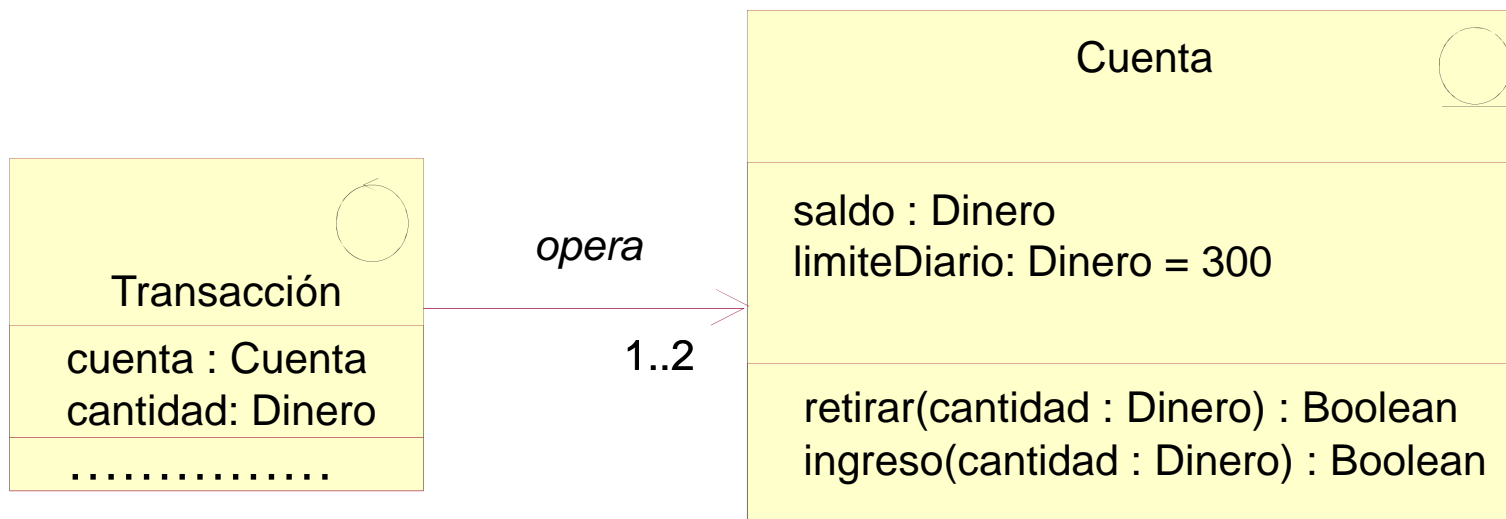




3.1.2 Artefactos. Clases de diseño

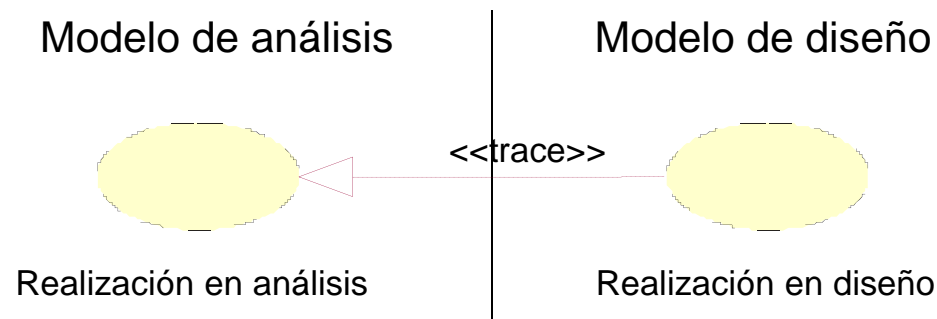
- Una clase de diseño es una abstracción de una clase de implementación
- Las operaciones, atributos, tipos, visibilidad (public, protected, private ...), etc se pueden especifican con la sintaxis del lenguaje elegido
- Las relaciones entre clases de diseño se traducen de manera directa al lenguaje:
 - generalización: herencia
 - asociaciones, agregaciones: atributos
- Se pueden postergar algunos requisitos a implementación (por ejemplo: manera de nombrar los atributos, operaciones, ...)
- Realizan interfaces.

3.1.2 Artefactos. Clases de diseño

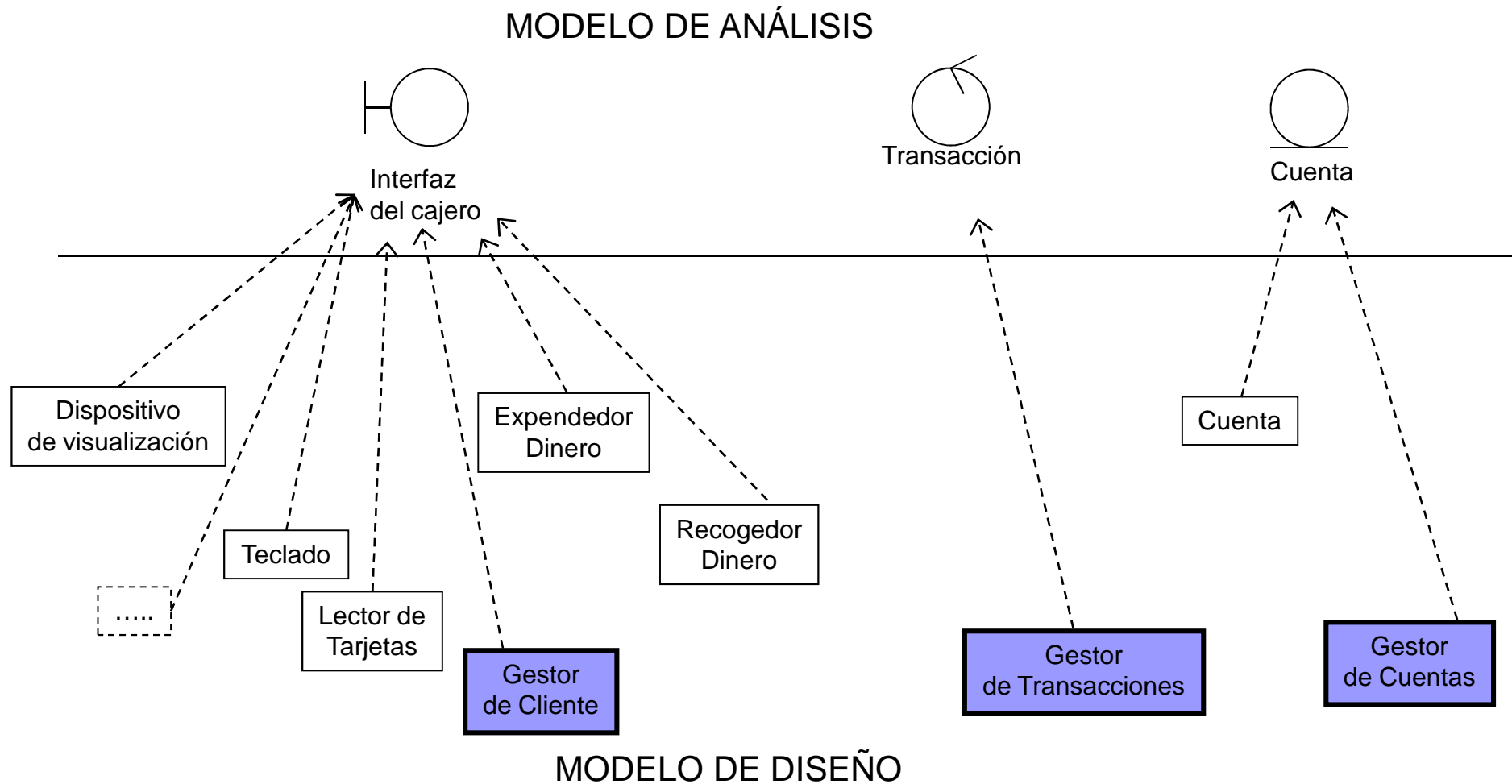


3.1.3 Artefactos. Realización en diseño de los casos de uso

- Es una colaboración que describe cómo se realiza en diseño un caso de uso en términos de clases de diseño y sus interacciones

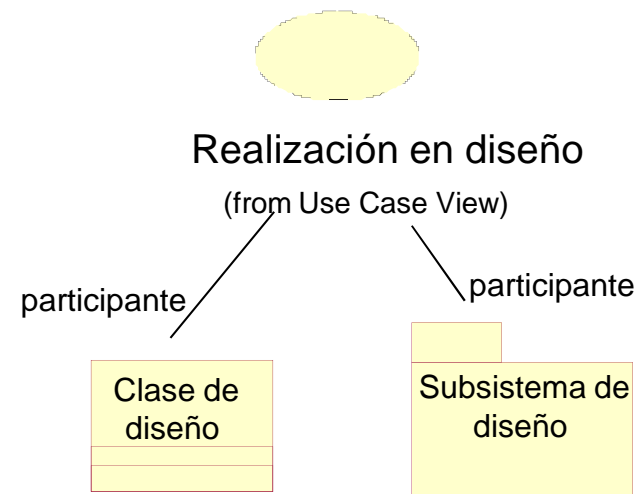


3.1.3 Artefactos. Realización en diseño de los casos de uso



3.1.3 Artefactos. Realización en diseño de los casos de uso

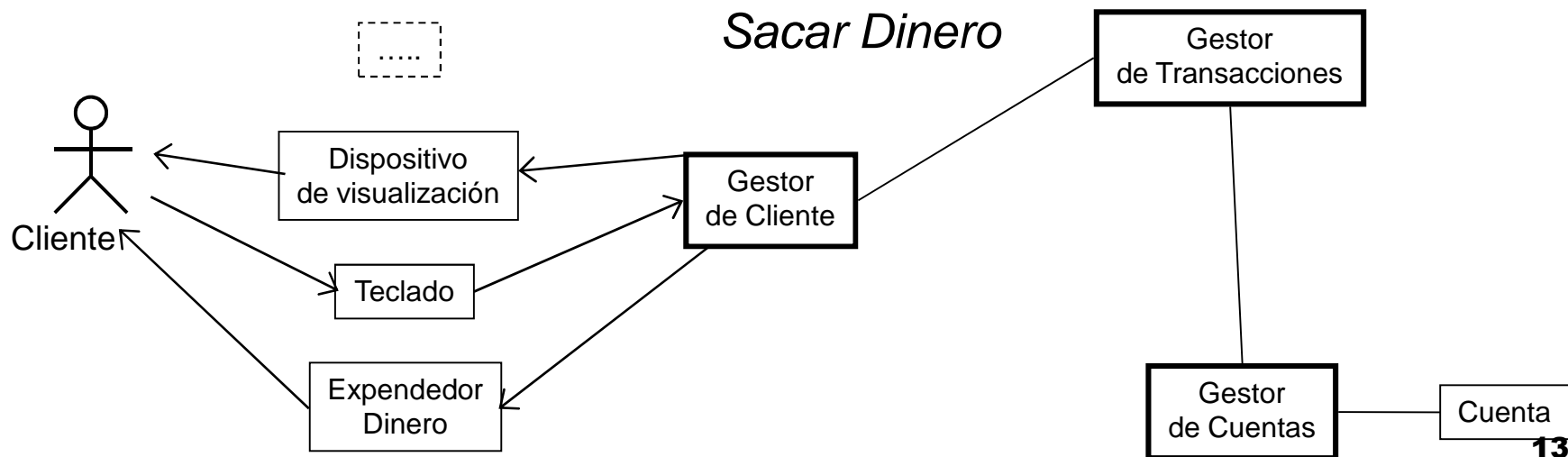
- La realización en diseño de un caso de uso, incluye:
 - diagramas de clases: clases participantes
 - diagramas de interacción: escenarios del caso de uso
 - descripción textual del flujo de eventos
 - Requisitos de implementación
 - Opcionalmente, subsistemas e interfaces



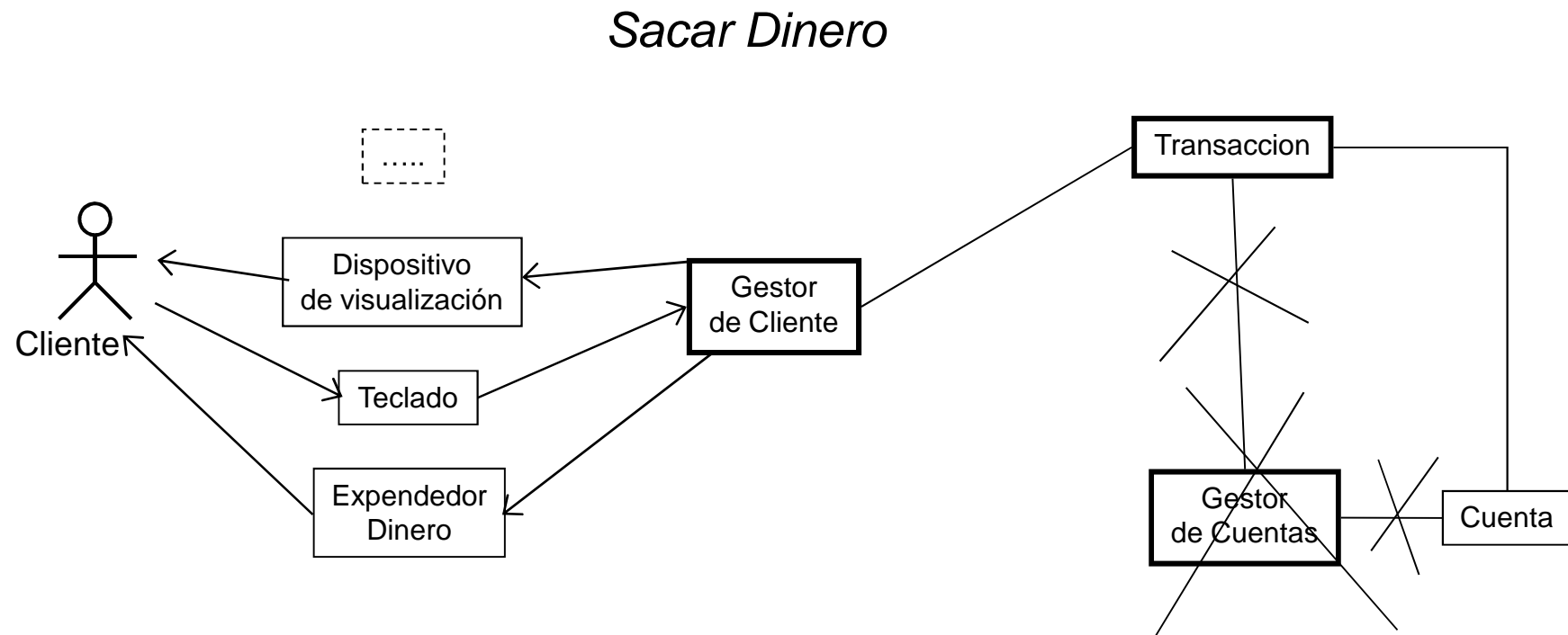
3.1.3 Artefactos. Realización en diseño de los casos de uso

Diagramas de clase

- Una clase de diseño puede participar en varios casos de uso
- Algunas responsabilidades, atributos y asociaciones suelen ser específicos de un sólo caso de uso.



3.1.3 Artefactos. Realización en diseño de los casos de uso





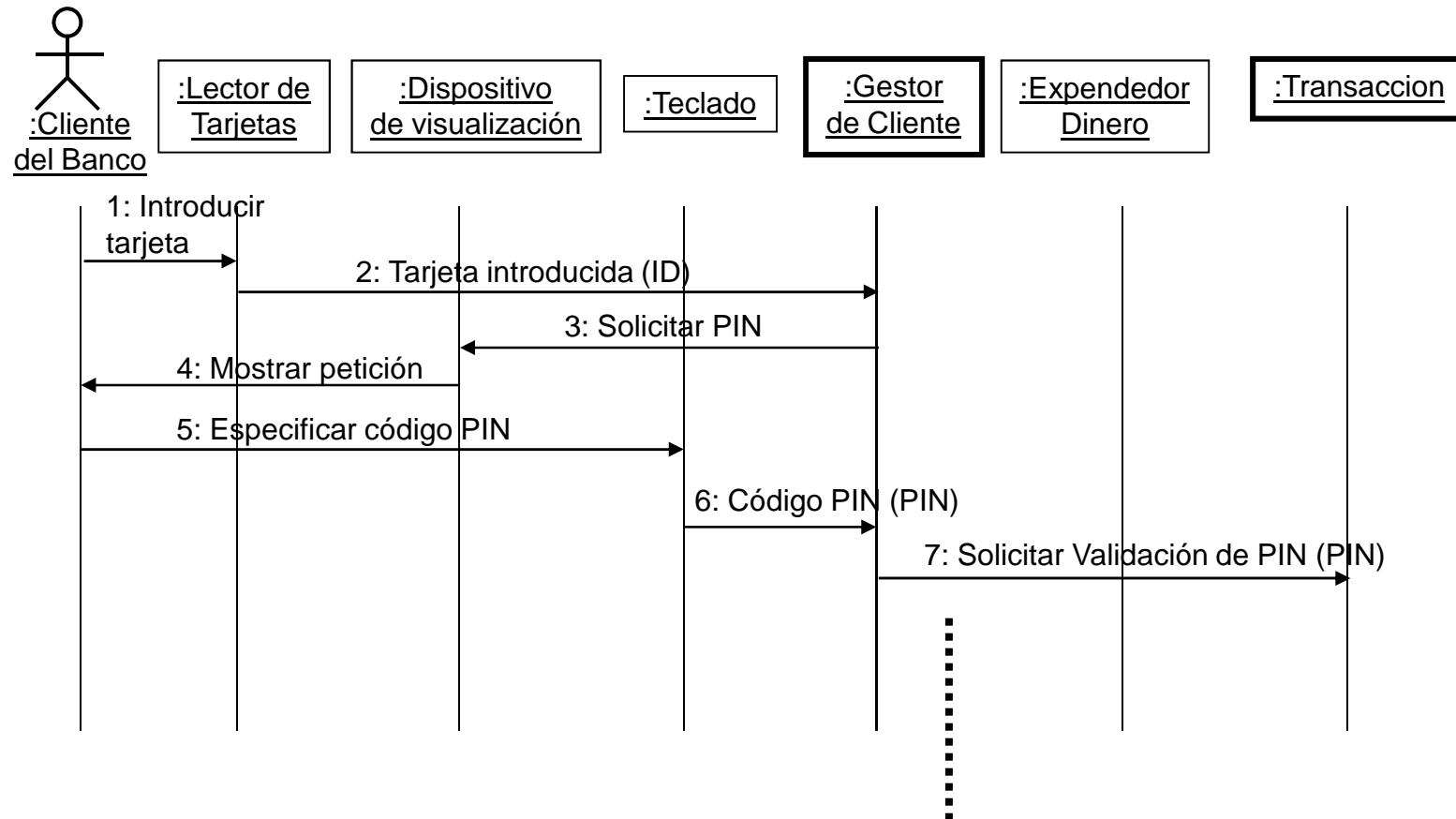
3.1.3 Artefactos. Realización en diseño de los casos de uso

Diagramas de interacción

- La secuencia de acciones en un caso de uso comienza cuando un actor envía un mensaje a un objeto de diseño.
- Utilizar mejor **diagramas de secuencia** que de colaboración. Nos interesa la secuencia cronológica de las interacciones.
- Se pueden incluir subsistemas y las interfaces que proporcionan

3.1.3 Artefactos. Realización en diseño de los casos de uso

Diagramas de interacción -> Diagramas de secuencia





3.1.3 Artefactos. Realización en diseño de los casos de uso

Flujo de eventos

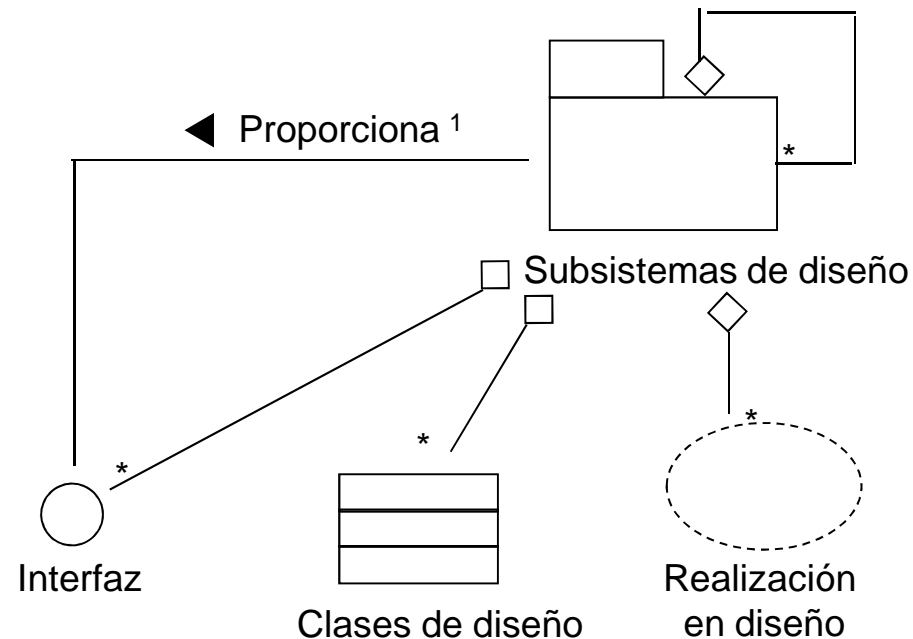
- Para clarificar los d. de secuencia: descripción textual
- Una descripción no tiene que ser local a un diagrama. Puede englobar a varios e indicar cómo se relacionan.

Requisitos de implementación

- Requisitos a gestionar en implementación
- Quizás durante esta fase de diseño se obtengan algunos nuevos
- Representarlos con restricciones {...} asignadas a las clases de diseño, operaciones, atributos, asociaciones, etc.

3.1.4 Artefactos. Subsistemas de diseño

- Forma de organizar los artefactos de diseño en piezas más manejables: clases de diseño, realización de casos de uso, interfaces y otros subsistemas.

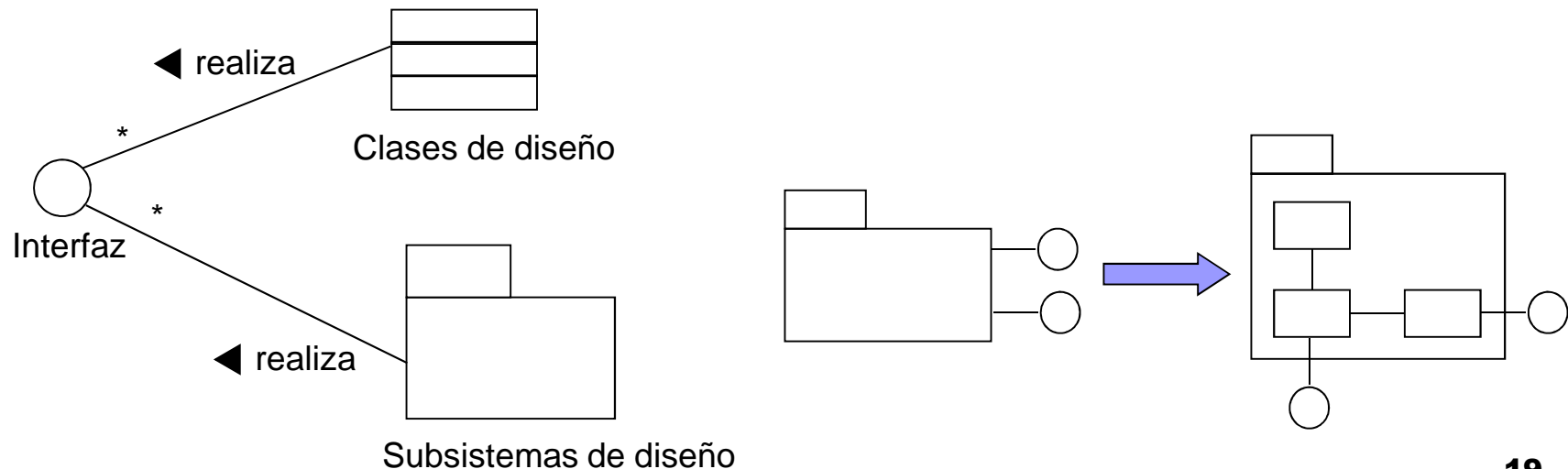


- Fuertemente cohesionados y débilmente acoplados.

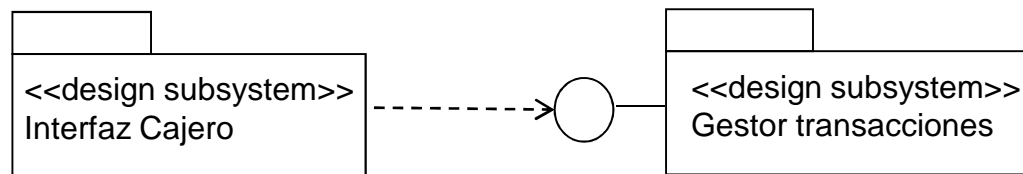
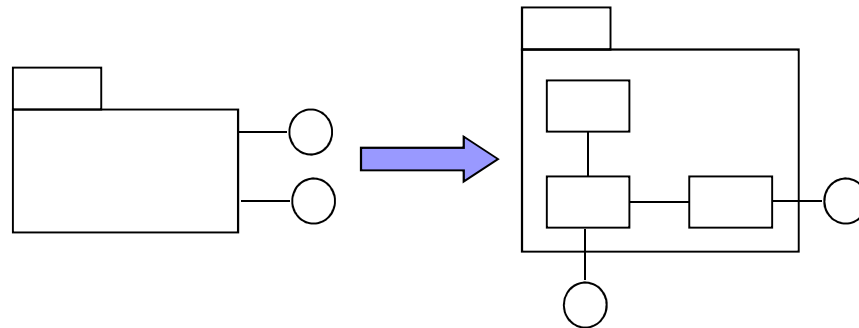
1.- representa la funcionalidad que exporta en términos de operaciones

3.1.5 Artefactos. Interfaz

- Los interfaces se utilizan para especificar las operaciones de las clases y los subsistemas de diseño
- Las clases de diseño soportan las operaciones de su interfaz mediante métodos.
- Los subsistemas de diseño soportan las operaciones de su interfaz mediante las clases de diseño (o subsistemas) que contiene.



3.1.5 Artefactos. Interfaz



Este subsistema "Gestor de transacciones" ofrece al subsistema "Interfaz Cajero" un conjunto de operaciones que pueden ser invocadas por el mismo, a través de la interfaz.



3. El diseño en el Proceso Unificado

3.1 Artefactos.

3.1.1 Modelo de diseño.

3.1.2 Clases de diseño.

3.1.3 Realización en diseño de los casos de uso.

3.1.4 Subsistemas en diseño.

3.1.5 Interfaz.

3.2 Actividades.

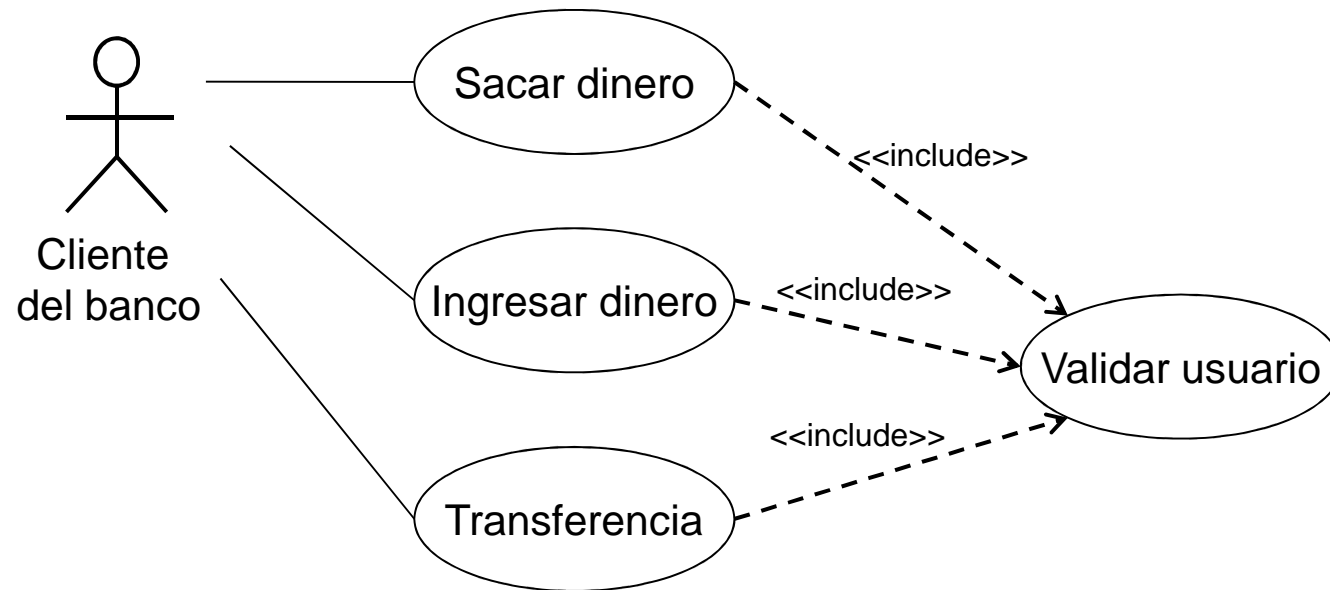
3.2.1. Diseño de los casos de uso.

3.2.2 Diseño de las clases.

3.2.3. Diseño de subsistemas.

3.2. Actividades

- Para ilustrar las actividades, utilizaremos el ejemplo del cajero automático





3.2.1 Actividades. Diseño de los casos de USO

2.1 Artefactos.

2.1.1 Modelo de diseño.

2.1.2 Clases de diseño.

2.1.3 Realización en diseño de los C.U.

2.1.4 Subsistemas

2.1.5 Interfaz

2.2 Actividades.

2.2.1. Diseño de los casos de uso.

2.2.2. Diseño de las clases.

2.2.3. Diseño de los subsistemas.

- Identificar las clases de diseño y/o subsistemas necesarios para la realización del caso de uso.
- Distribuir el comportamiento del caso de uso entre las clases y/o subsistemas de diseño.

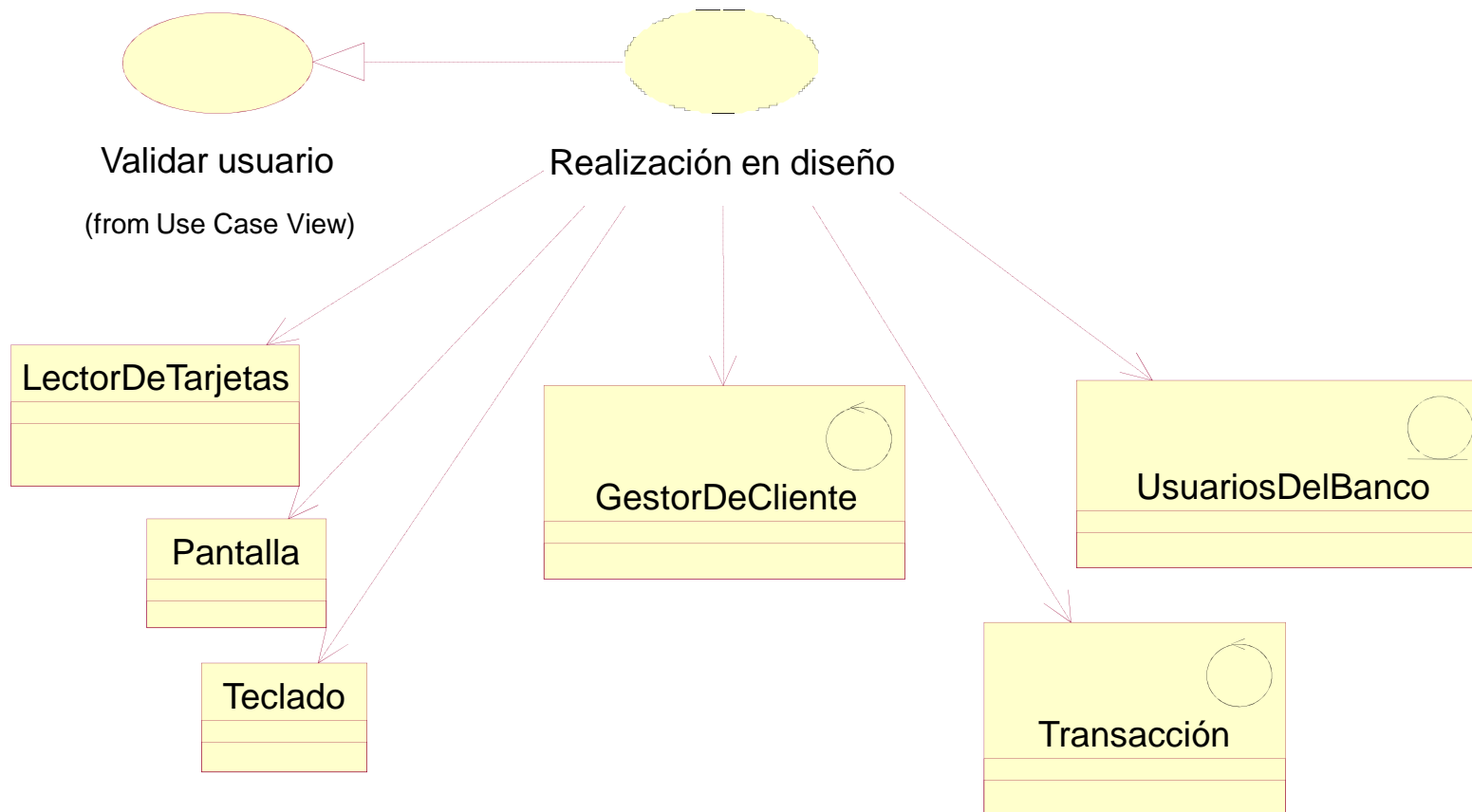


3.2.1 Actividades. Diseño casos de uso

3.2.1.1 Identificar las clases de diseño

- Derivar las clases de diseño de las correspondientes clases de análisis que participan en el caso de uso.
- Estudiar los requisitos especiales del caso de uso: realizarlos con los mecanismos genéricos de diseño o con clases de diseño.
- Asignar responsabilidades a las clases identificadas.
- Realizar un diagrama de clases que muestre las clases de diseño que intervienen en la realización del caso de uso y las relaciones entre ellas.

Diseño del caso de uso: “Validar usuario”





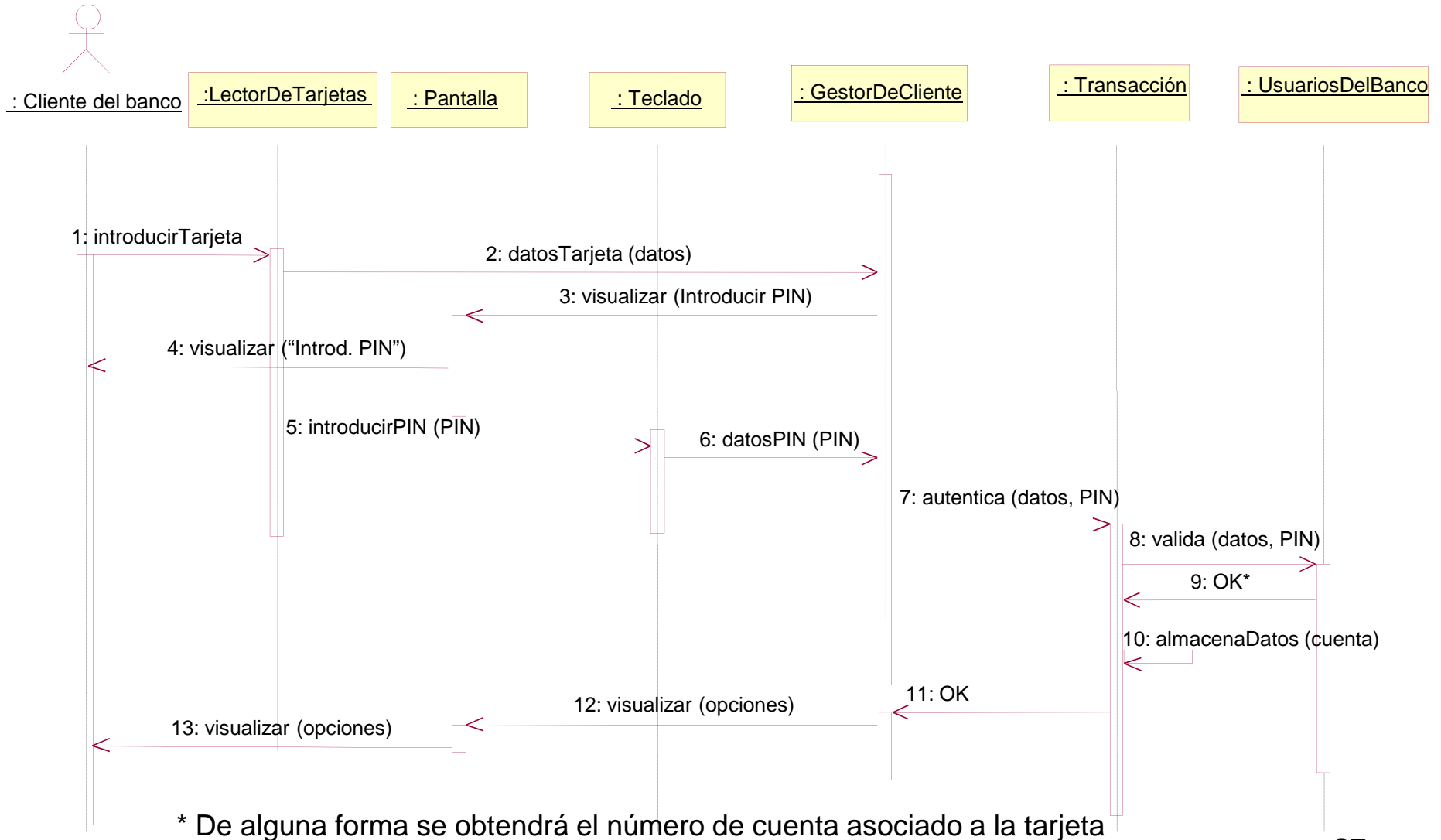
3.2.1 Actividades. Diseño casos de uso

3.2.1.2 Describir interacciones entre objetos de diseño

- Utilizar diagramas de secuencia
 - objetos, instancias de actores, enlaces
- Crear un diagrama de secuencia
- Comenzar estudiando la realización en análisis del c.u.
- Sobre los diagramas de secuencia:
 - el caso de uso comienza cuando una instancia de un actor envía un mensaje a un objeto interfaz.
 - cada clase de diseño identificada debería tener al menos un objeto participando en el diagrama de secuencia.
- En esta fase gestionar excepciones y errores (entradas incorrectas, situaciones anormales, etc.)

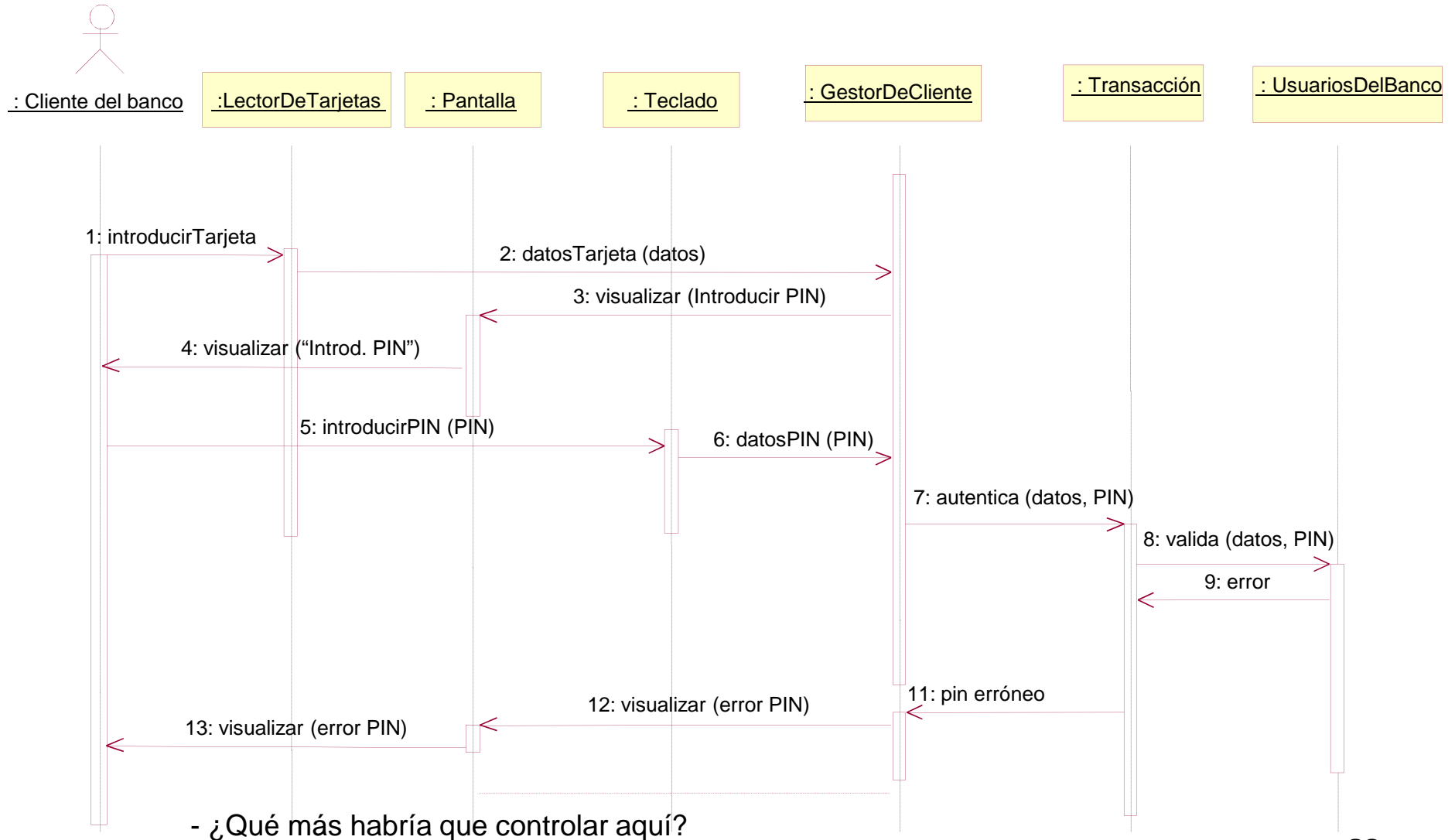
Diseño del caso de uso: "Validar usuario"

Secuencia correcta



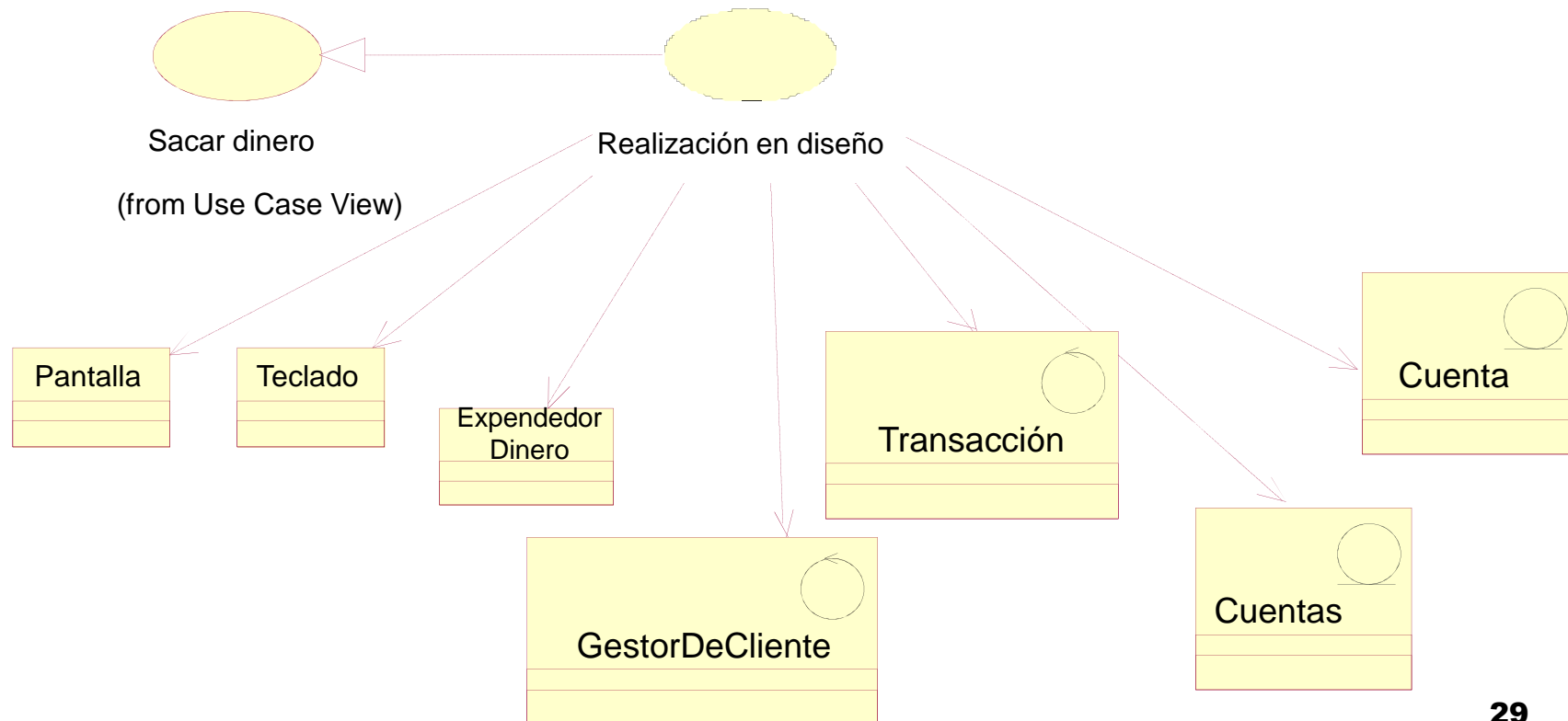
Diseño del caso de uso: "Validar usuario"

Camino alternativo: Código incorrecto



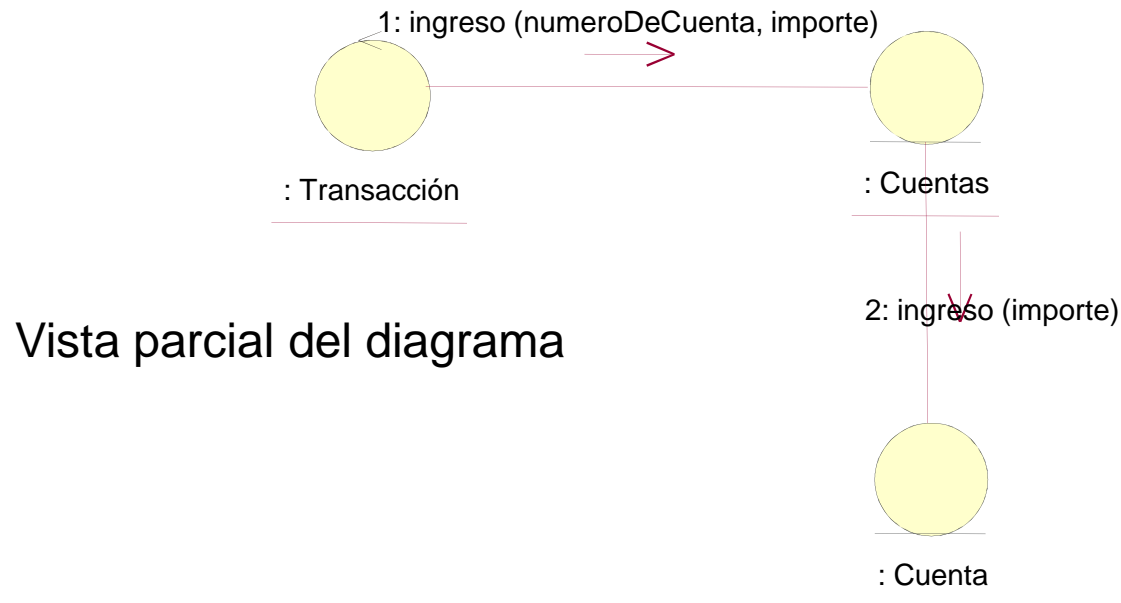
Diseño del caso de uso: “Sacar dinero”

- Suponemos que el usuario ya ha sido identificado (se ha ejecutado el caso de uso anterior).
- Ahora selecciona la opción “sacar dinero”.

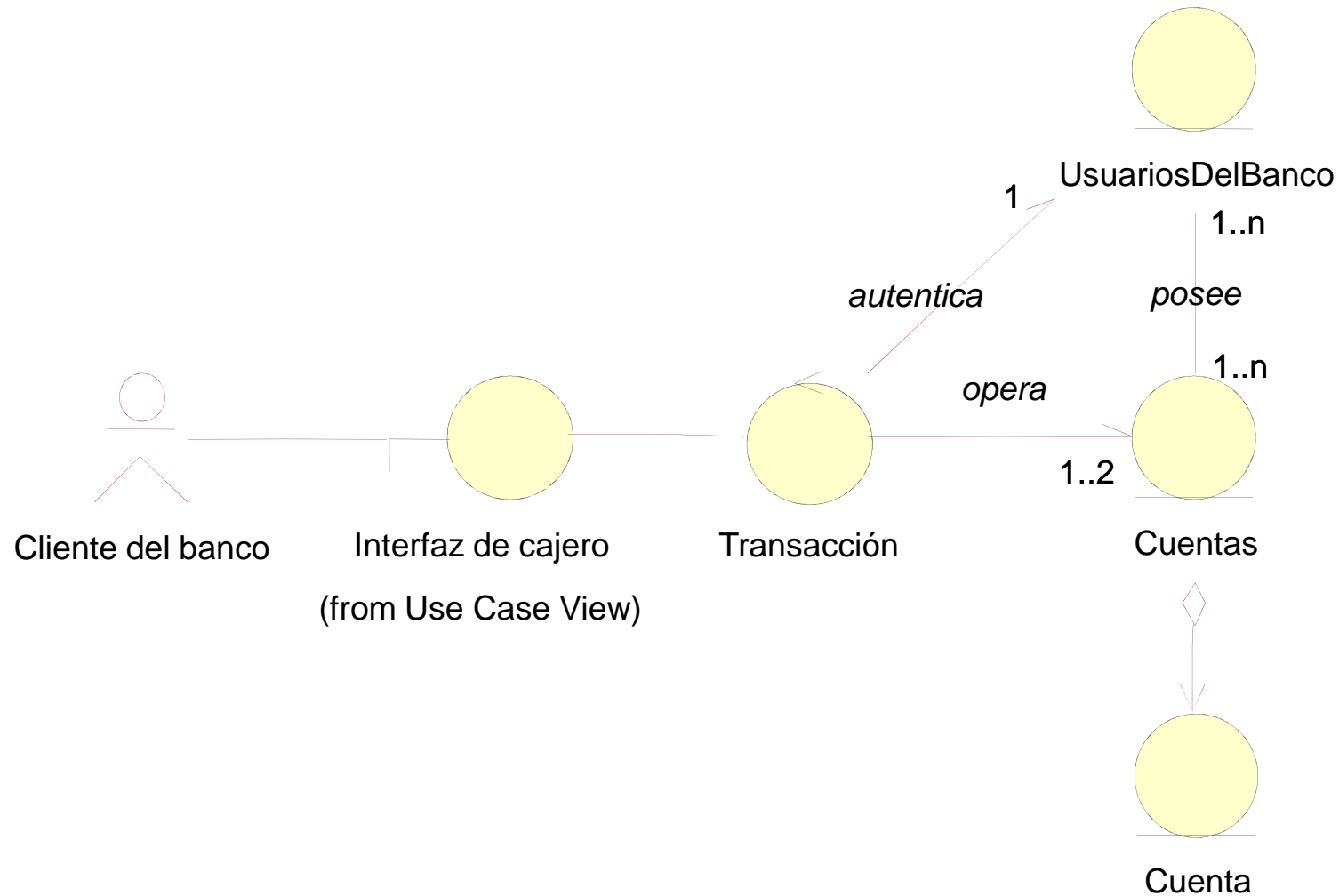


Refinando el caso de uso: “Sacar dinero”

- Añadimos la clase **Cuentas** que asocia número de cuenta con una instancia de la clase **Cuenta**.
- La clase **Transacción** ya no actuará directamente sobre **Cuenta**.

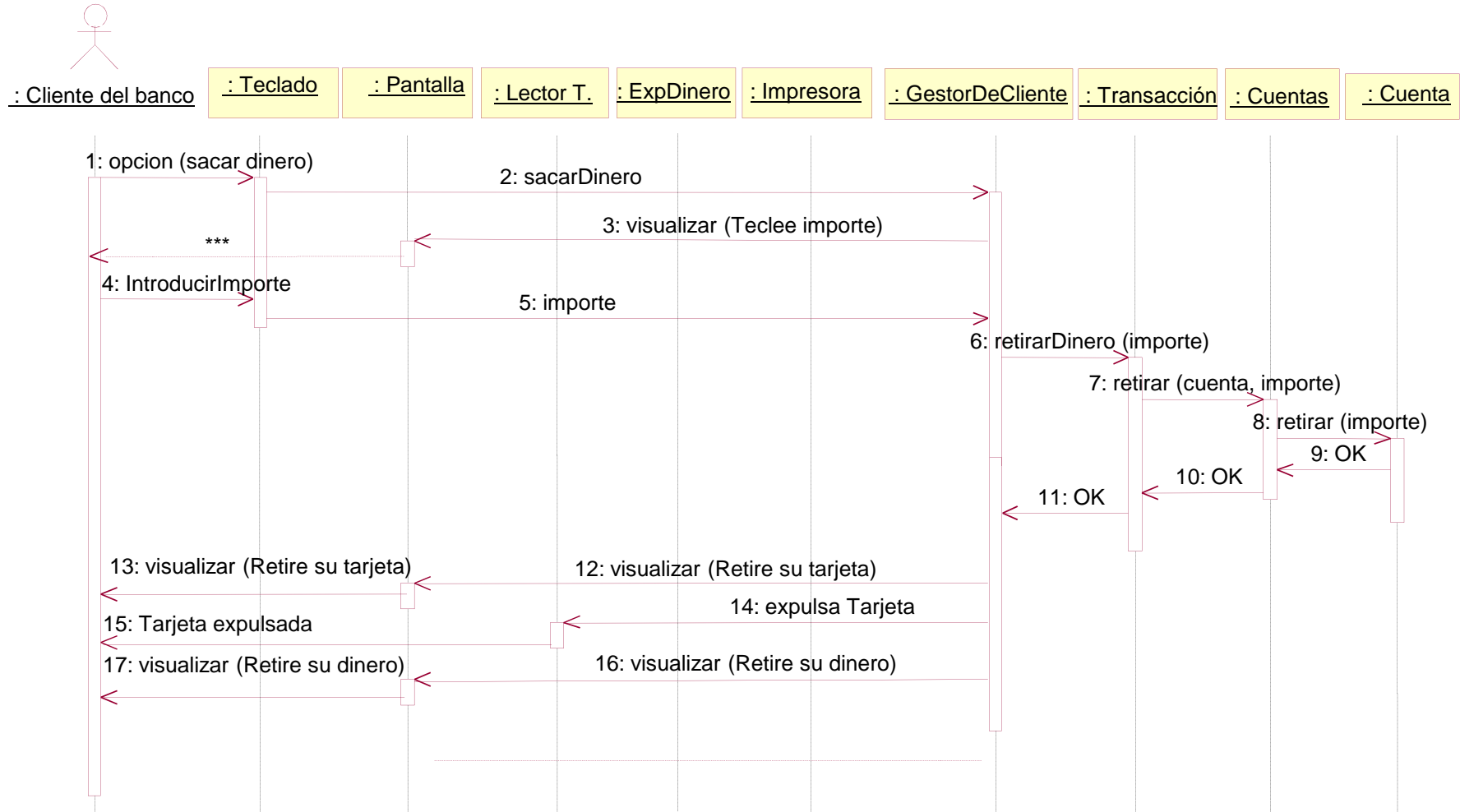


Refinando el caso de uso: "Sacar dinero"



Diseño del caso de uso: "Sacar dinero"

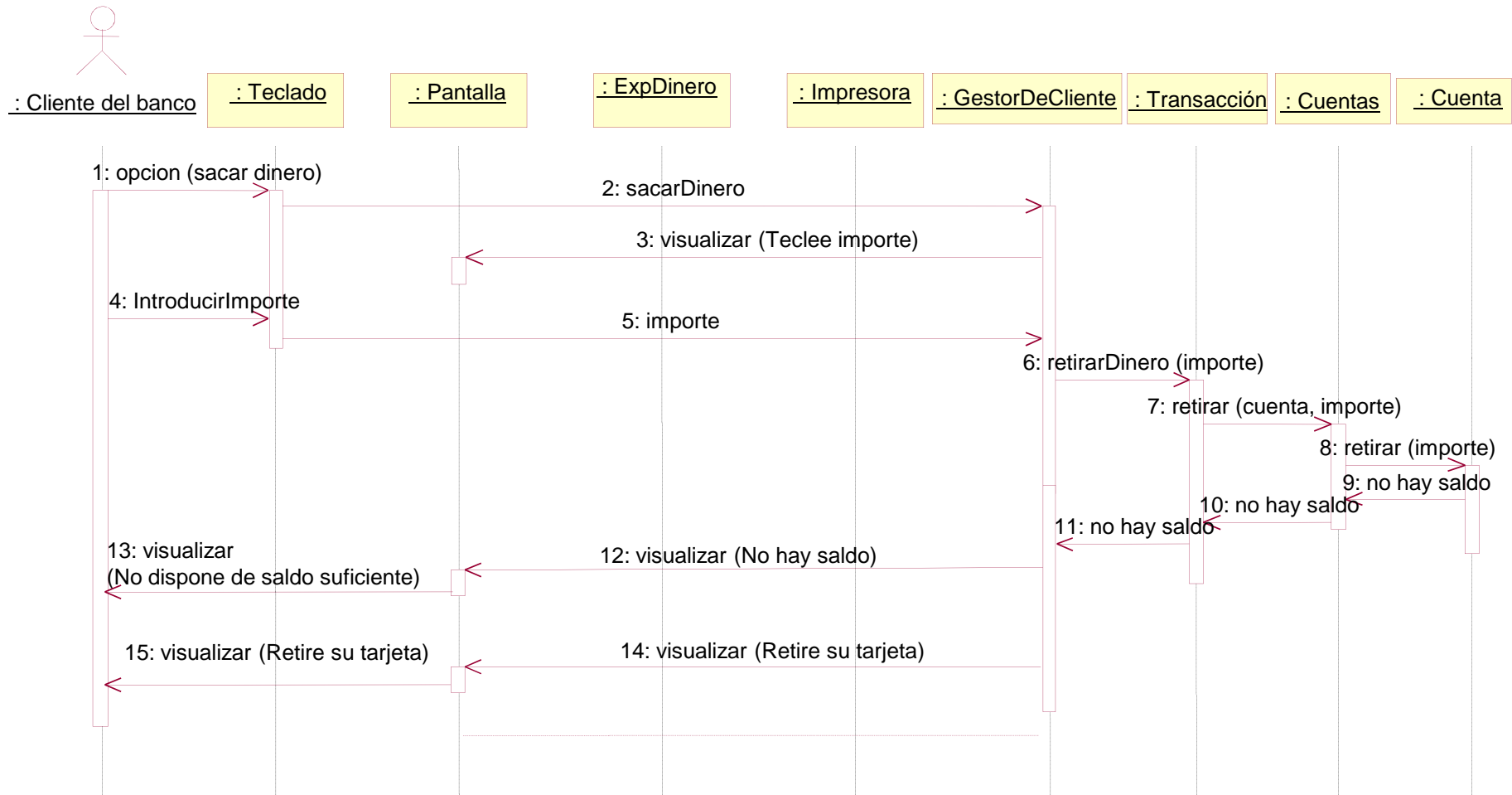
Secuencia correcta



*** Se pueden obviar estos mensajes
¿Qué pasa con la impresora?

Diseño del caso de uso: "Sacar dinero"

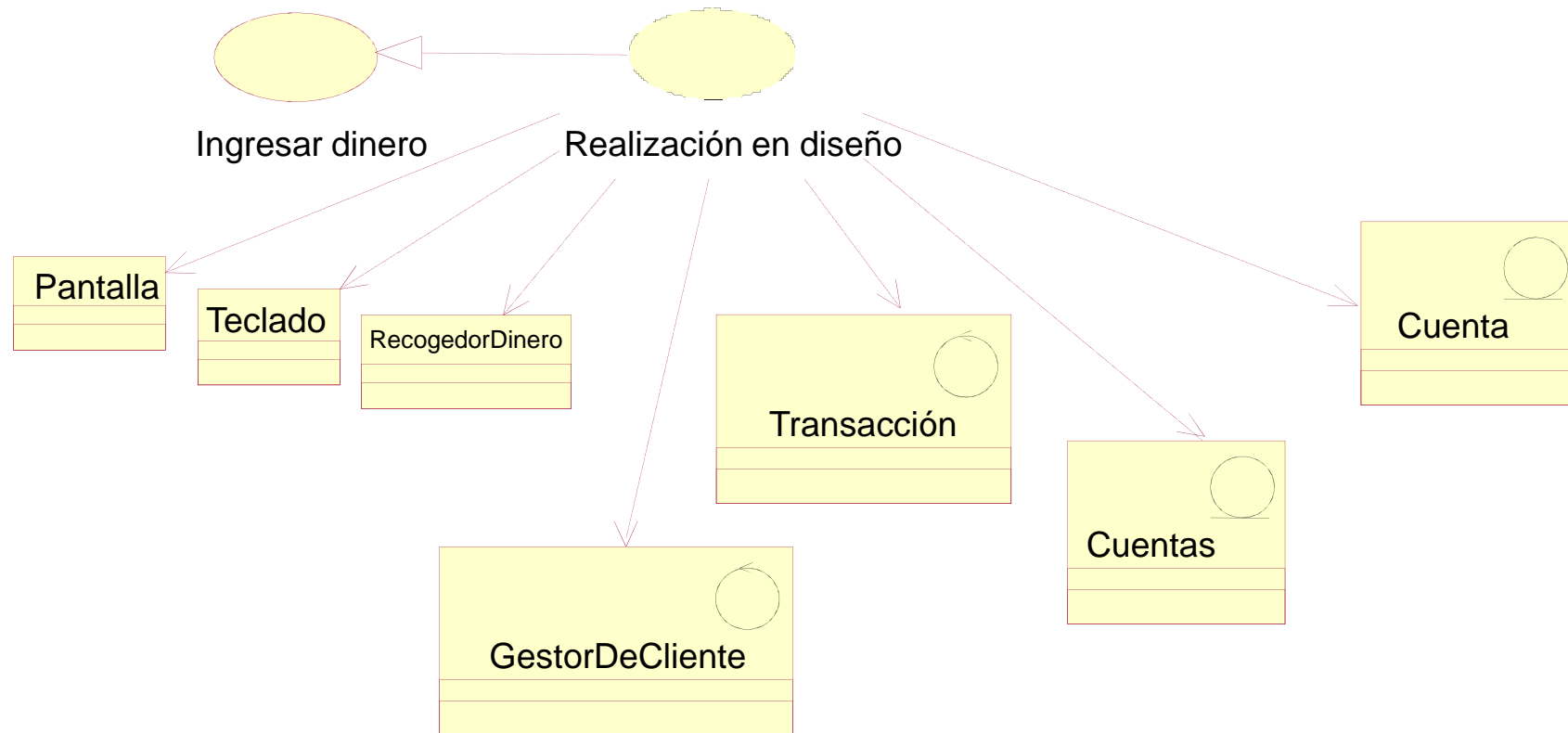
No hay fondos



Falta:

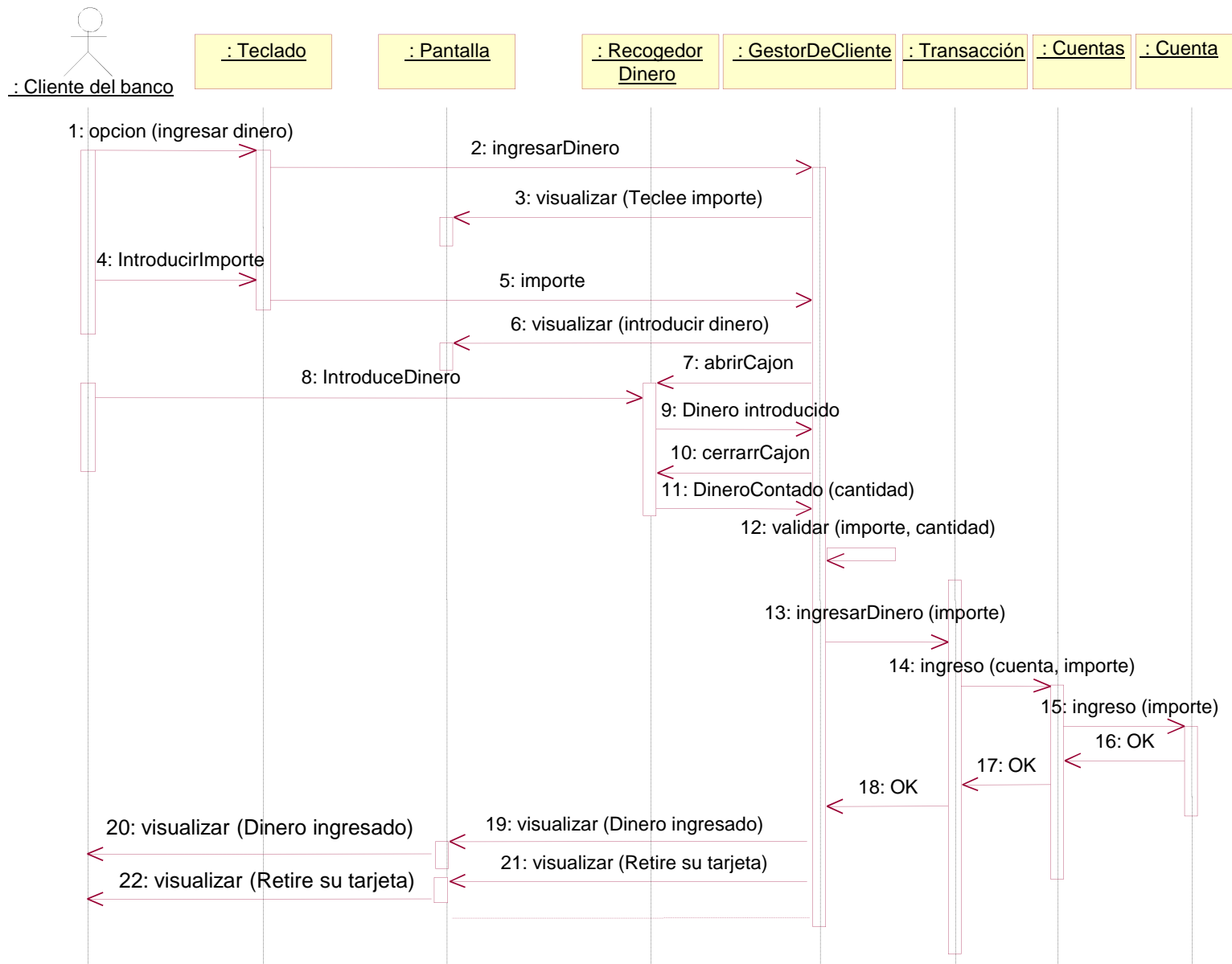
- se ha superado el límite diario
- en el cajero no hay dinero

Diseño del caso de uso: “Ingresar dinero”



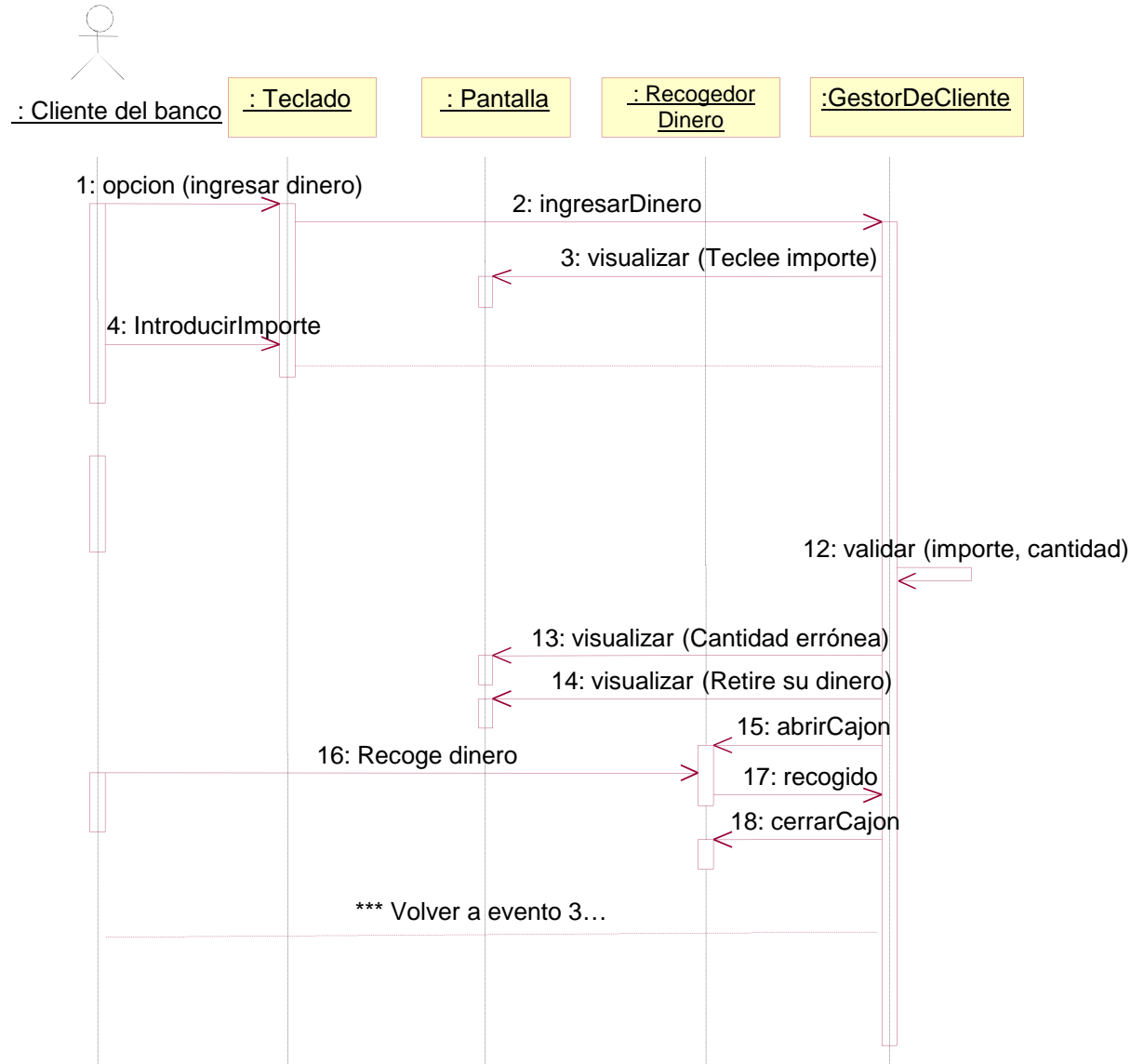
Diseño del caso de uso: "Ingresar dinero"

Secuencia correcta



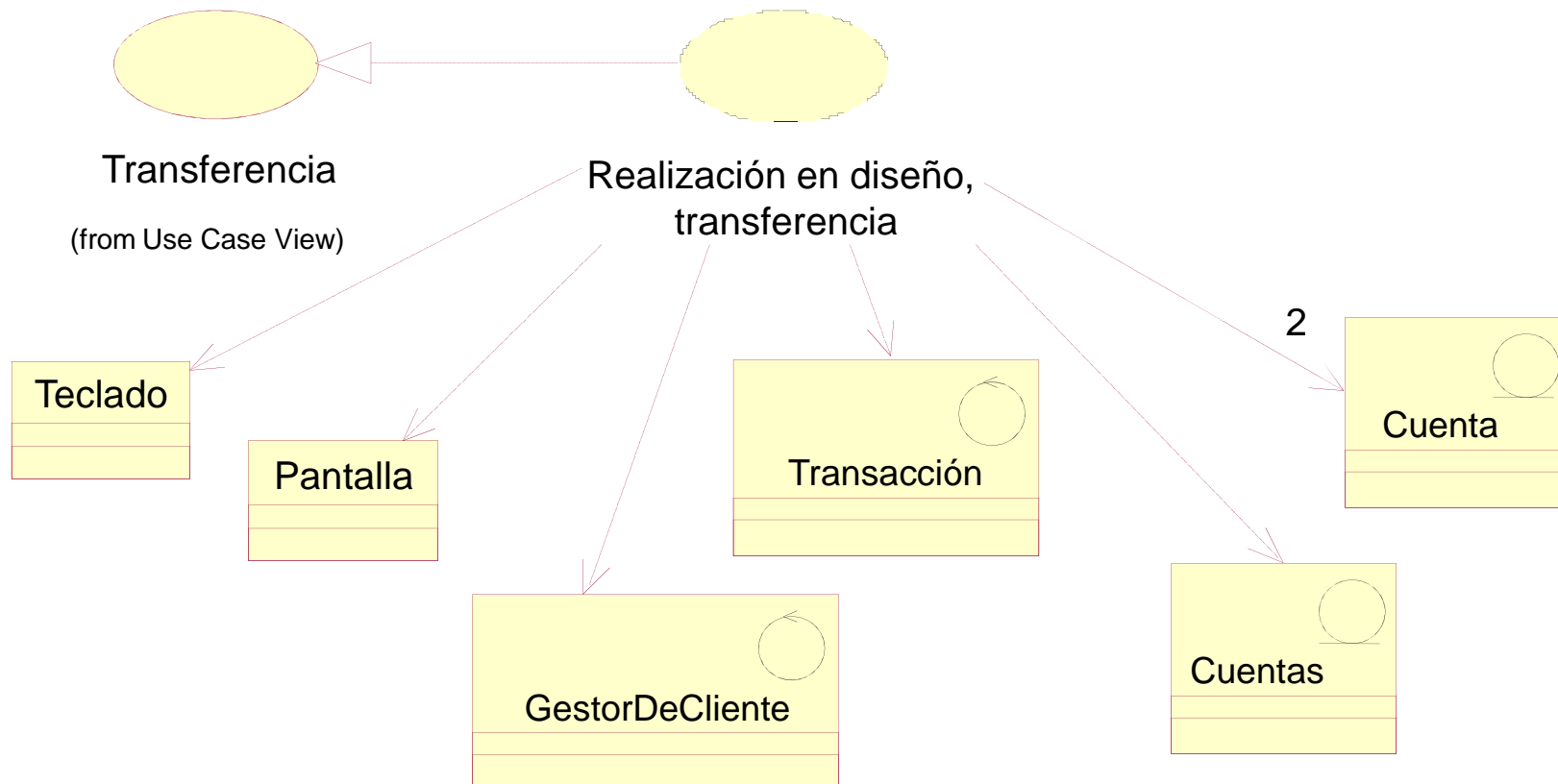
Diseño del caso de uso: "Ingresar dinero"

Cantidad incorrecta



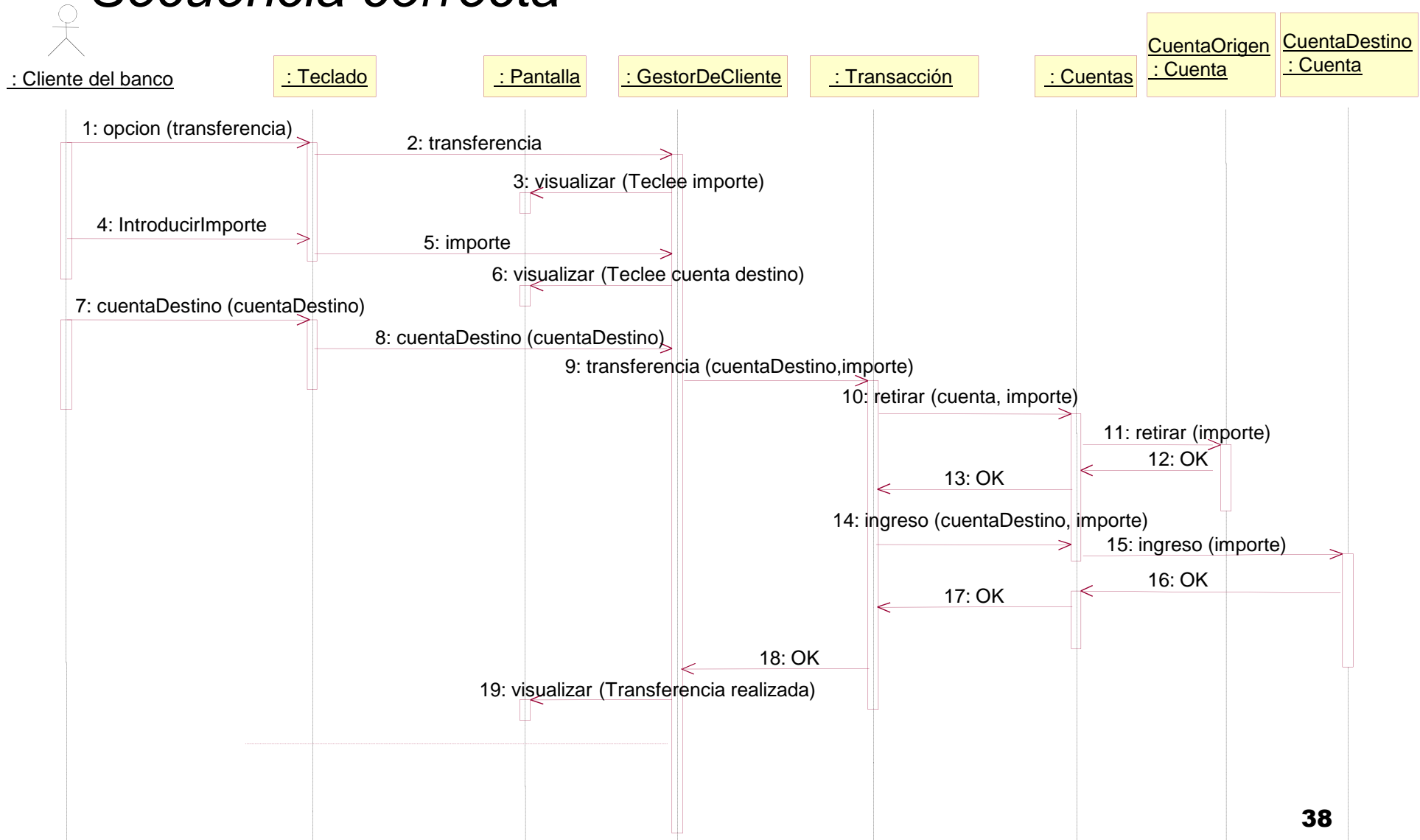
Diseño del caso de uso: “Transferencia”

- Suponemos que el usuario ya ha sido identificado (se ha ejecutado el caso de uso anterior). Ahora selecciona la opción “transferencia”.



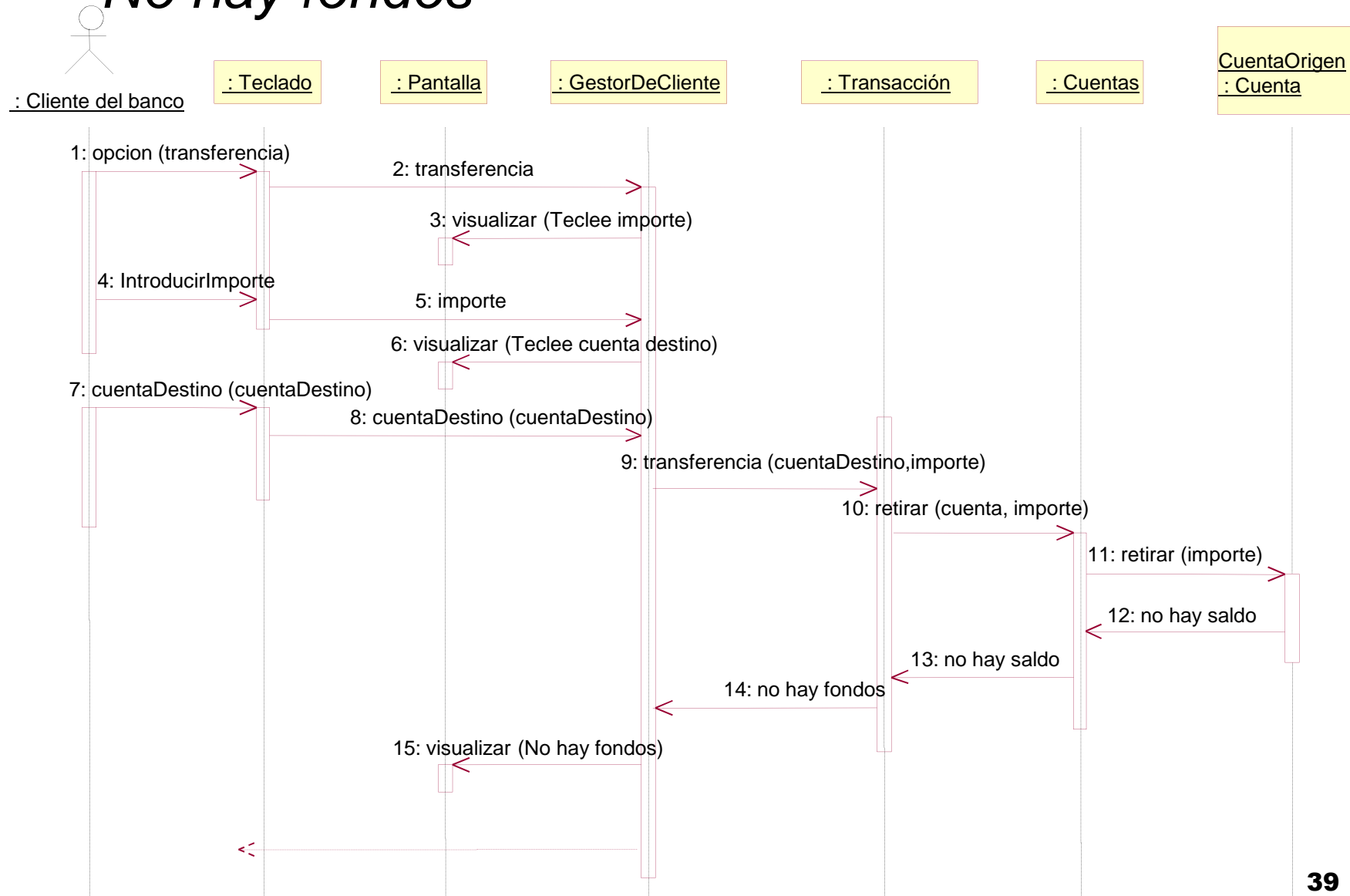
Diseño del caso de uso: "Transferencia"

Secuencia correcta

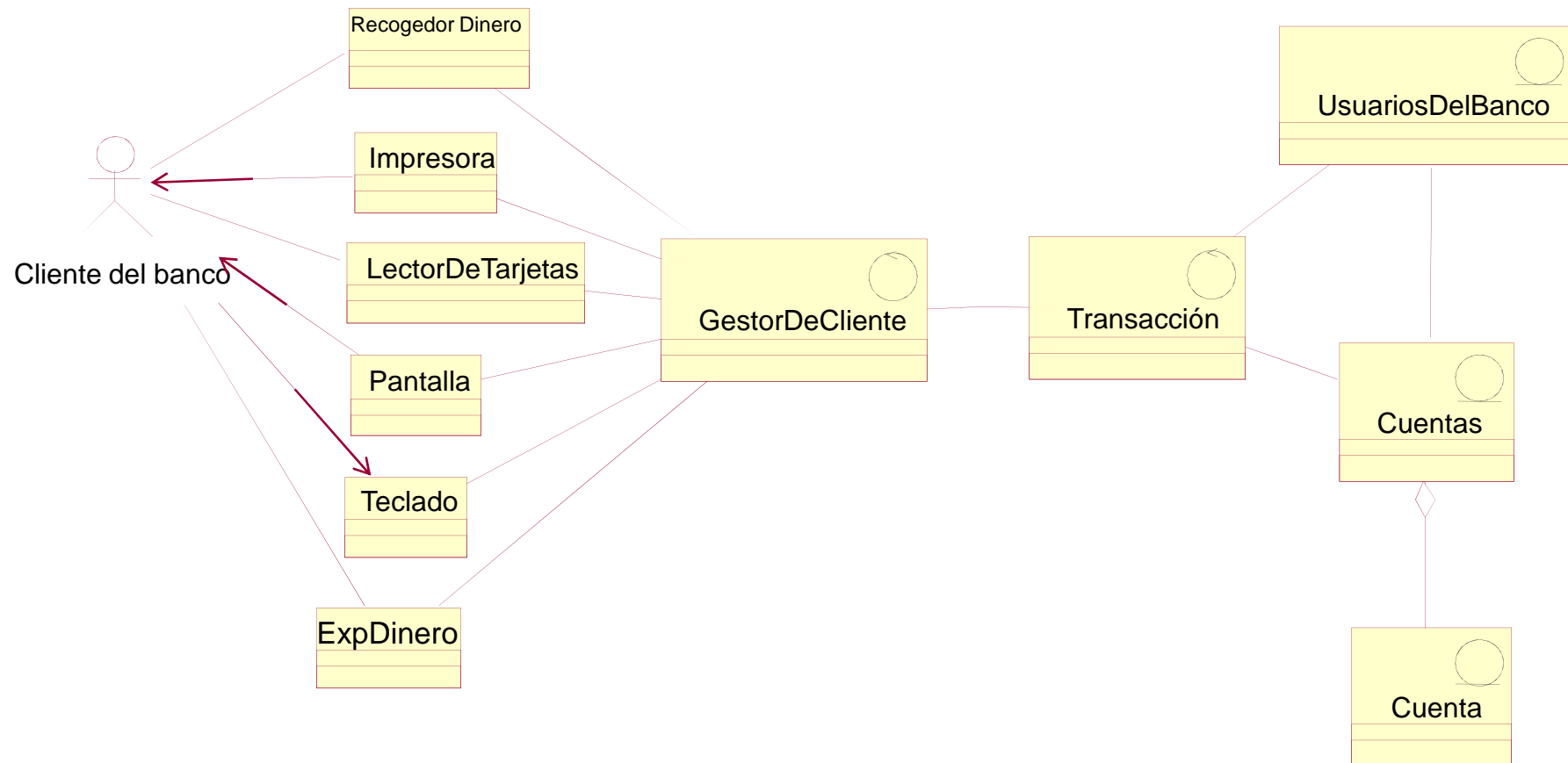


Diseño del caso de uso: "Transferencia"

No hay fondos



Modelo de clases de diseño





3.2.2 Actividades. Diseño de las clases

2.1 Artefactos.

2.1.1 Modelo de diseño.

2.1.2 Clases de diseño.

2.1.3 Realización en diseño de los C.U.

2.1.4 Subsistemas

2.1.5 Interfaz

2.2 Actividades.

2.2.1. Diseño de los casos de uso.

2.2.2. Diseño de las clases.

2.2.3. Diseño de los subsistemas.

- Identificar las responsabilidades de las clases de diseño (papeles en los casos de uso)
- Identificar:
 - operaciones
 - atributos
 - relaciones en las que participa
 - estados (diagramas de estados)
 - métodos que soportan sus operaciones
 - Requisitos nuevos...
- ¿¿Quién tiene el control de la ejecución??



3.2.2. Actividades. Diseño de las clases

3.2.2.1 Identificar operaciones

- En el lenguaje de implementación
- Mirar responsabilidades que tiene en los casos de uso

3.2.2.2 Identificar atributos

- Describirlos en el lenguaje de programación
- Considerar los atributos de las clases de análisis de las que se derivan



3.2.2. Actividades. Diseño de las clases

3.2.2.3 Identificar asociaciones y agregaciones

- Las interacciones en los diagramas de secuencia precisan de asociaciones entre las clases que interactúan.
- Minimizar el número de relaciones entre clases (disminuir el acoplamiento).
- Refinar multiplicidad, papeles, etc.
- Refinar la navegabilidad (dirección) de las asociaciones en base a los diagramas de secuencia.
- Identificar generalizaciones-especializaciones.



3.2.2. Actividades. Diseño de las clases

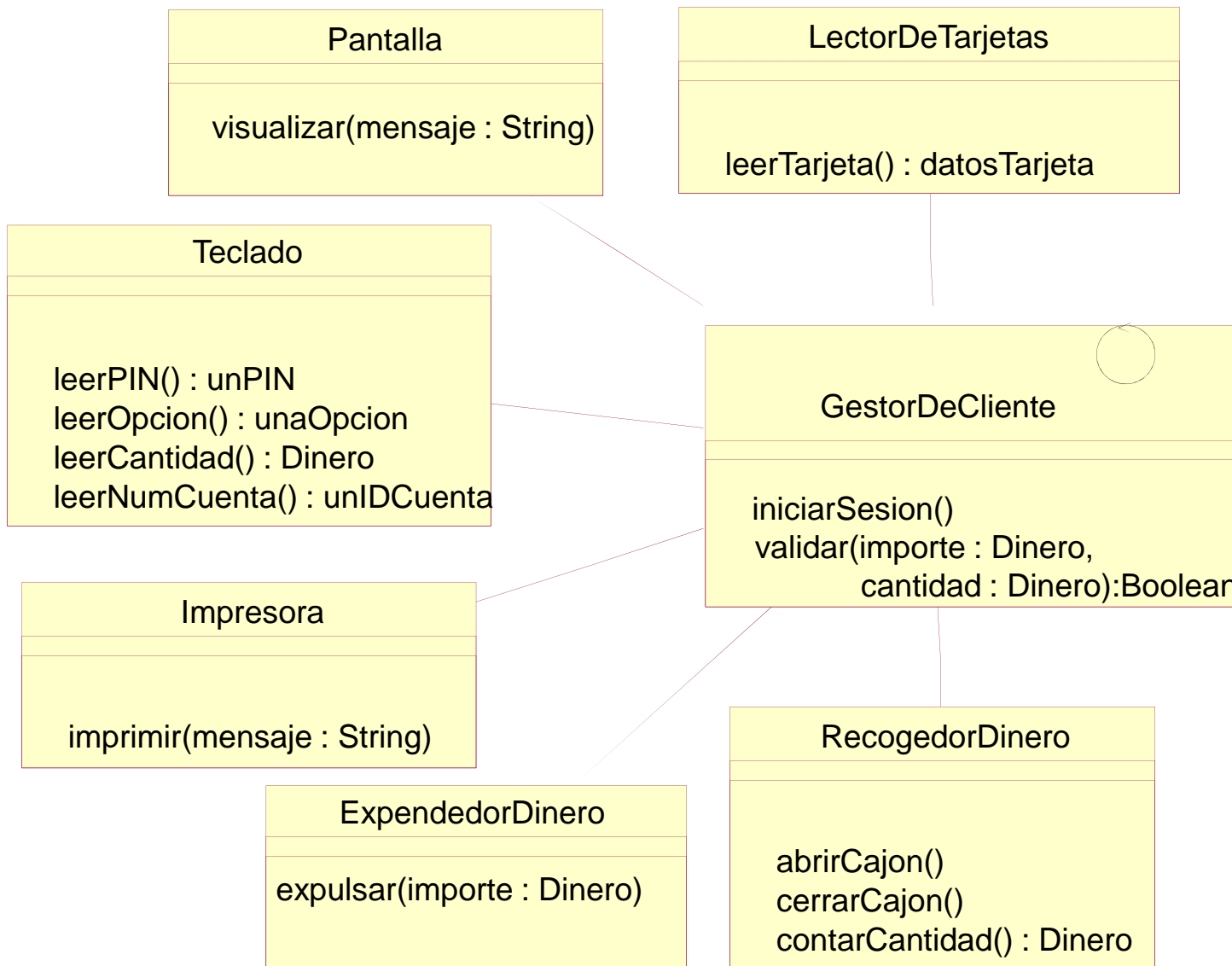
3.2.2.4 Describir métodos

- Algoritmos para implementar alguna operación (lenguaje natural).
- Esqueletos de métodos generado por la herramienta.
- En general, en lenguaje de programación **se suele hacer en implementación.**

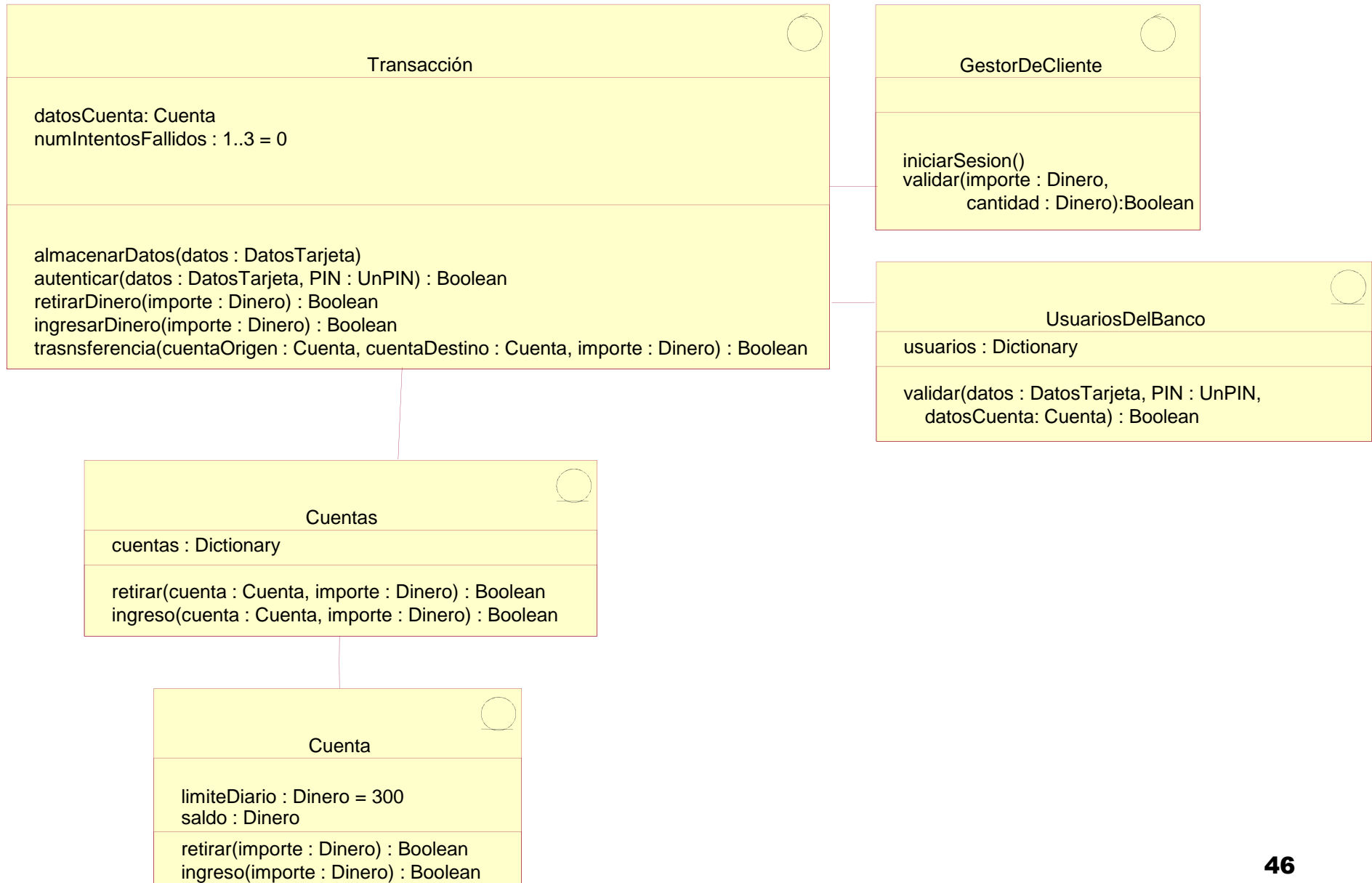
3.2.2.5 Describir estados

- Algunos objetos reaccionan en función de su estado actual. Utilizar diagramas de transición de estados.

Modelo de clases de diseño



Modelo de clases de diseño

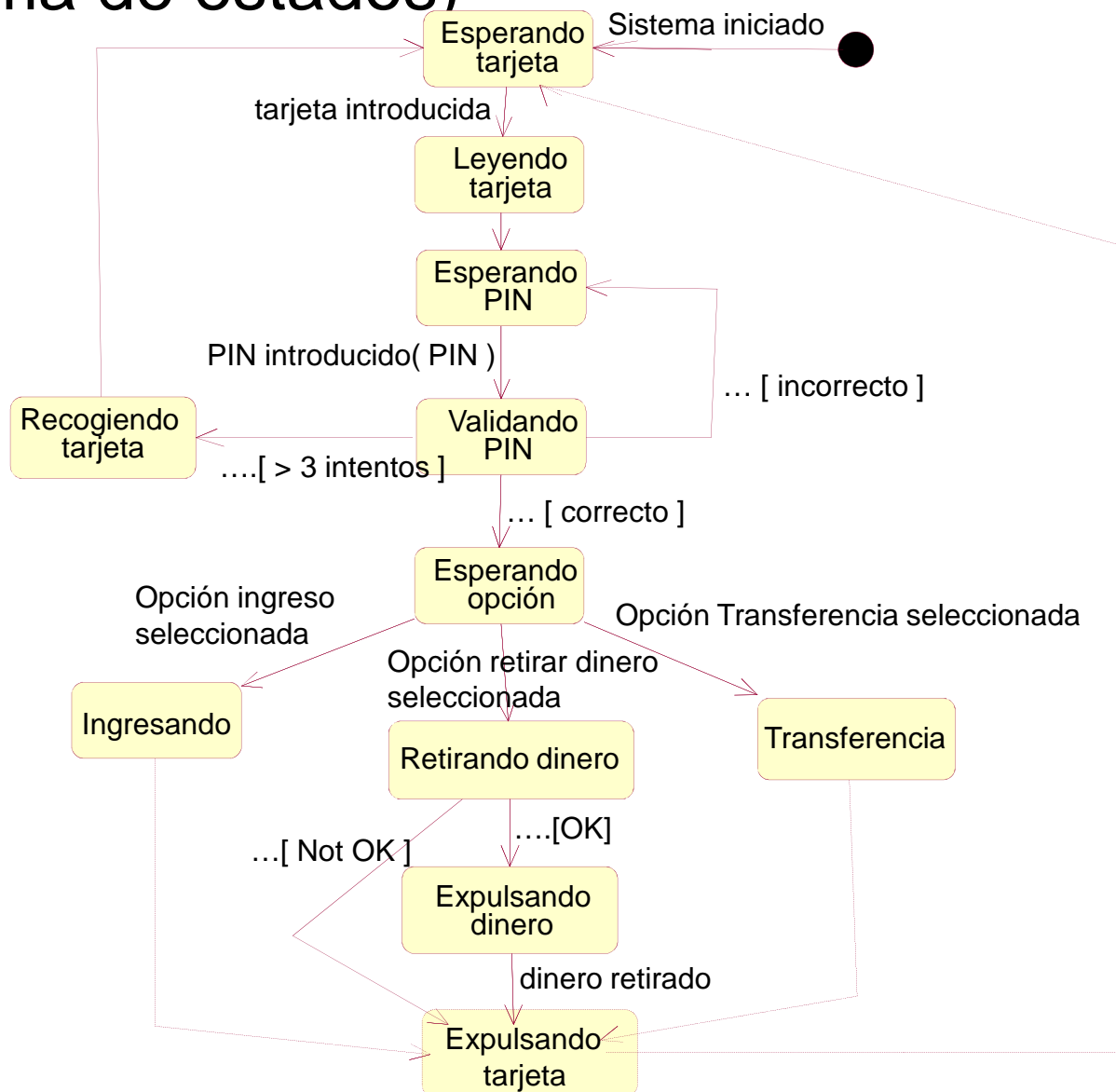





Diseño de la clase “GestorDeCliente”

- En una aplicación orientada a objetos debe existir una clase que represente la propia aplicación. Este sería el punto donde comenzaría la ejecución de la misma.
- La única clase que tiene un comportamiento parecido a la propia aplicación sería **GestorDeCliente**. Su comportamiento se puede representar mediante una máquina de estados. Realizar el diagrama de transición de estados del sistema Cajero Automático.

Diseño de la clase “GestorDeCliente” (boceto de diagrama de estados)





3.2.3. Actividades. Diseño de los subsistemas

- Intentar que los subsistemas de diseño estén débilmente acoplados.
- Intentar que las clases dentro de los subsistemas tengan una alta cohesión.
- Describir las dependencias entre los subsistemas.
- Determinar qué clases de unos subsistemas interactúan con qué otras clases de otros subsistemas.
- Asegurarse que el subsistema soporta sus interfaces.
- Objetivos:
 - Subsistemas independientes
 - Garantizar corrección de interfaces
 - Garantizar la realización de dichas interfaces