

UML: Lenguaje Unificado de Modelado



Tema 4

Introducción

- Similitud:
 - Arquitectos, edificios, planos
 - Ing. Inf., programas, diagramas
- UML
 - Unified Modeling Language. Versión 2.0 (finales 2004)
 - Diagramas (ing. inf.)
 - Usados como esquemas y menos con información rigurosa (“planos de arquitectos”)
 - Dos modos:
 - Ingeniería inversa: a partir de código hacer diagramas
 - Ingeniería directa: hacer diagramas y luego implementar
- Dominio
 - Mundo en el que hay definido un problema
- Modelo:
 - Abstracción de un problema
 - Formado por objetos

Indice

- **Diagramas de Casos de Uso.**
- Diagramas de Estructura.
- Diagramas de Comportamiento.
- OCL.
- Herramientas.
- Ejemplos.
- Bibliografía.

Casos de Uso

- Describen qué hace el sistema desde el punto de vista de un observador externo.
- Ponen énfasis en **qué** hace el sistema, no en **cómo** lo hace.
- Un escenario es una instancia particular de un diagrama de casos de uso.
 - Ejemplo de lo que ocurre cuando alguien interactúa con el sistema

Casos de Uso

- **Actor** = Algo con comportamiento (persona, otro programa, organización...), que interactua con el sistema.
- **Escenario** (instancia de caso de uso) = Secuencia de acciones e interacciones entre los actores y el sistema.
- **Caso de Uso** = Colección de escenarios (éxito y fracaso) que describen actores que usan el sistema para conseguir un objetivo.

Casos de Uso

- Pasos:

- Identificar los límites del sistema.
- Identificar los actores principales.
- Para cada uno, identificar sus objetivos.
- Definir casos de uso que satisfagan sus objetivos.

Ejemplo

Aplicación para una Galería de Arte

Te encargan realizar una aplicación para la compra-venta de cuadros. En cuanto a la compra de cuadros, una vez que el agente introduce unos datos básicos sobre el cuadro, el sistema debe proporcionar el precio recomendado que el agente de la galería debería pagar. Si el vendedor del cuadro acepta la oferta, entonces el agente de la galería introduce más detalles (sobre el vendedor del cuadro y la venta).

Los datos básicos incluyen el nombre y apellidos del artista, el título y fecha de la obra, sus dimensiones, la técnica (óleo, acuarela u otras técnicas), el tema (retrato, naturaleza muerta, paisaje, otro) y la clasificación (obra maestra, obra representativa, otro tipo). Si es obra maestra, el precio recomendado se calcula comparando el cuadro introducido con los que hay en el registro de cuadros, tomando el más parecido y aplicando un algoritmo que tiene en cuenta la coincidencia de tema, la técnica y las dimensiones del cuadro. El sistema debe utilizar información de subasta de todo el mundo que ahora la galería recibe en un CD de manera mensual. Para una obra representativa, el precio recomendado se calcula como si fuera una obra maestra y luego se aplica una corrección. Para una obra de otro tipo, se calcula utilizando el área del cuadro y un coeficiente de moda para el artista. Si no hay coeficiente de moda para un artista, el agente tiene por norma no comprar el cuadro. El coeficiente de moda varía de mes a mes.

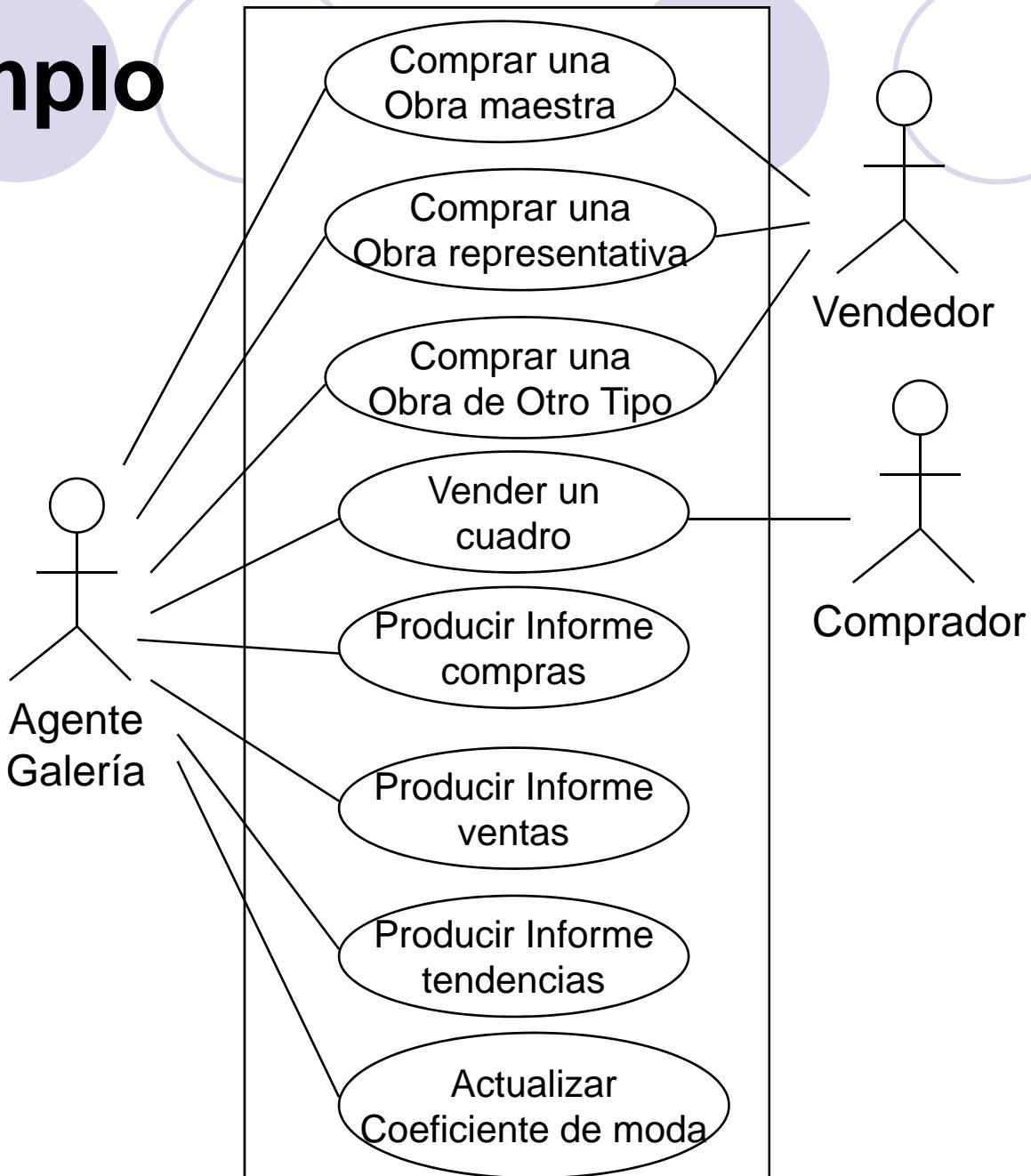
Si el cuadro finalmente se compra, se introducen datos adicionales.

En cuanto a la venta de cuadros por parte de la galería, el sistema simplemente registra la fecha de venta, el nombre y dirección del comprador y el precio de venta real.

El sistema también deberá detectar nuevas tendencias en el mercado de arte tan pronto como sea posible. La idea es detectar secuencias de compras por valores mayores que los esperados por la obra de un artista determinado, de tal manera que tu cliente pueda comprar cuadros de ese artista antes de que otros detecten la tendencia. Con el objetivo de detectar cuándo el precio de venta es mayor que el precio esperado cuando tu cliente compró el cuadro, se debe mantener un registro de todas las compras y todas las ventas.

Se quieren generar tres informes: compras y ventas realizadas durante un año, y artistas de moda....

Ejemplo



Ejemplo

Caso de uso: comprar una obra maestra

Actores primarios:

Agente Galería, vendedor

Interesados y Objetivos:

- **Agente**: quiere obtener una recomendación lo más acertada posible del precio máximo recomendado de manera rápida.
- **Vendedor**: quiere vender el cuadro a un precio razonable de manera rápida.

Precondiciones:

El agente ha entrado en la aplicación.

Garantía de éxito (post-condiciones):

Se registra la venta.

Escenario Principal de Éxito:

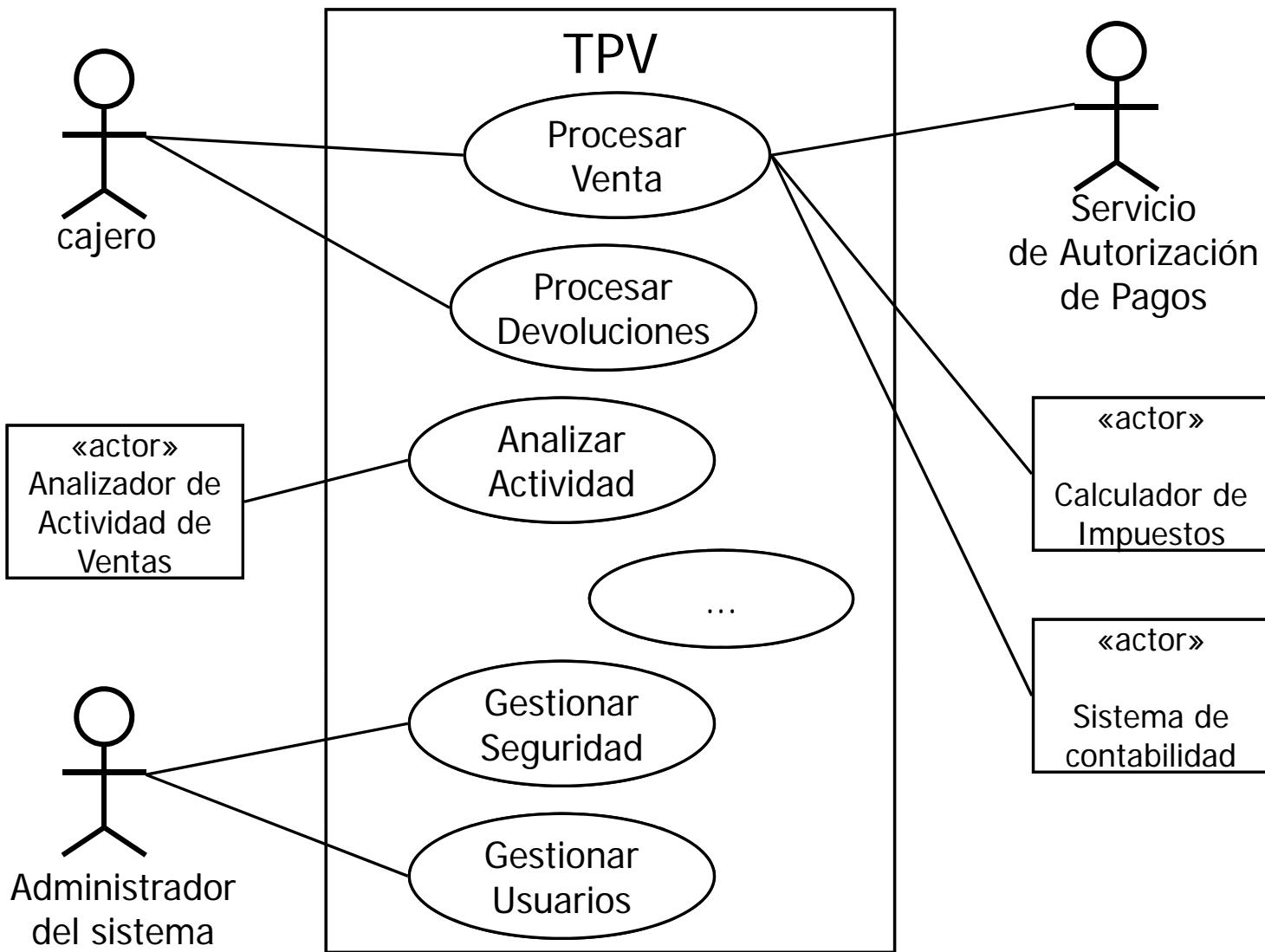
1. El agente introduce la descripción del cuadro.
2. El sistema busca el cuadro más parecido del mismo autor.
3. El sistema presenta el precio recomendado.
4. El agente hace una propuesta por debajo del precio recomendado, y el vendedor acepta la oferta.
5. El agente introduce información de la venta.

Extensiones:

- 2a. No hay ningún cuadro parecido del mismo autor, así que el sistema no presenta una recomendación.
- 4a. El vendedor no acepta la oferta y la venta no se produce.

Ejemplo

Terminal Punto de Venta



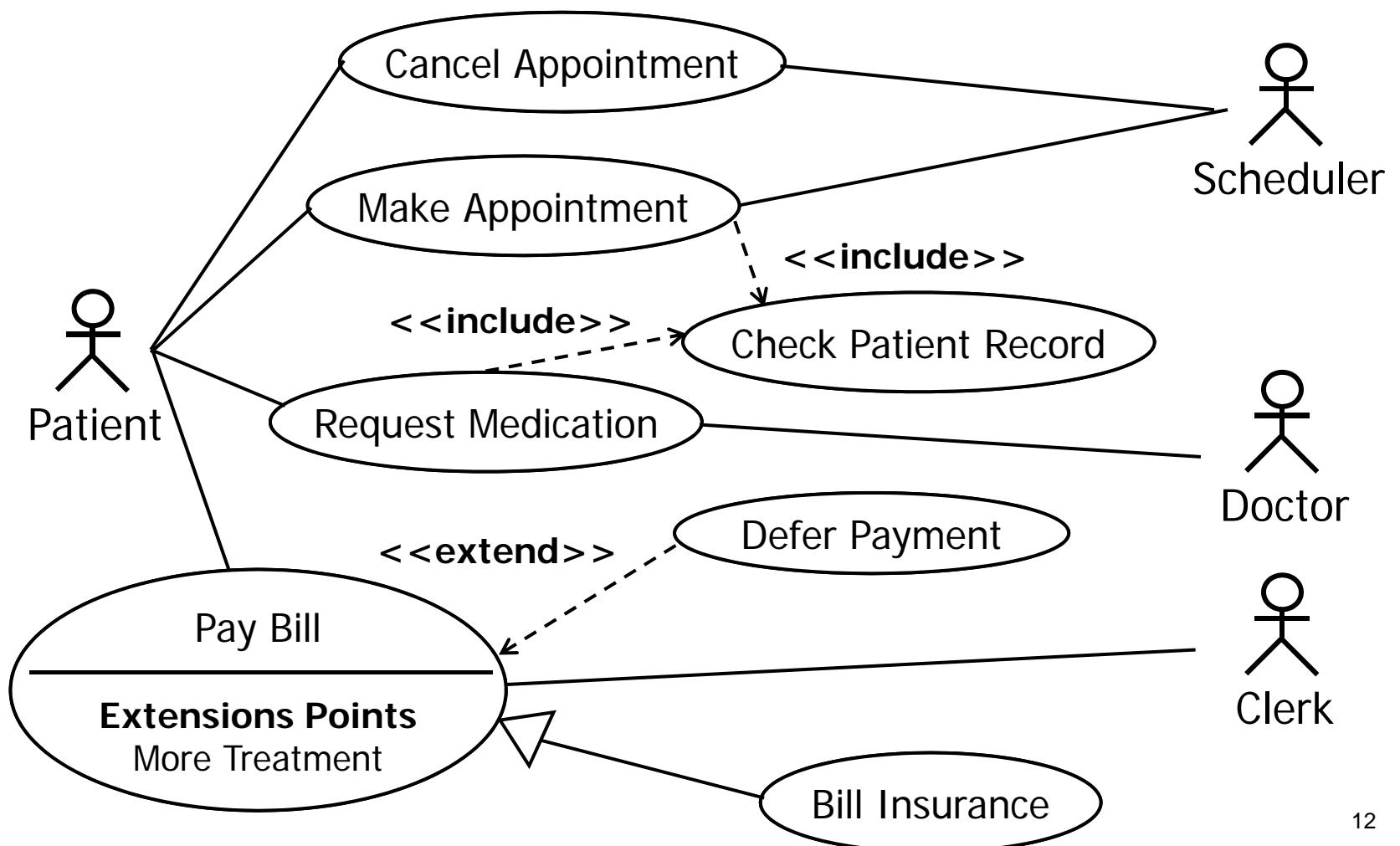
Diagramas de Caso de Uso

Relaciones

- Relaciones entre dos casos de uso
 - Generalización
 - “Es un caso particular de ...”
 - Herencia de clases en POO (overriding)
 - También se puede tener esta relación entre dos actores
 - Inclusión (*include*)
 - “Implica hacer también ...”
 - Extensión (*extend*)
 - “Se insertará en ...” un determinado punto (llamado “punto de extensión”) dependiendo de una condición.
 - Si un caso de uso depende de varios casos de uso mediante “extend” tendrá un punto de extensión para cada uno.

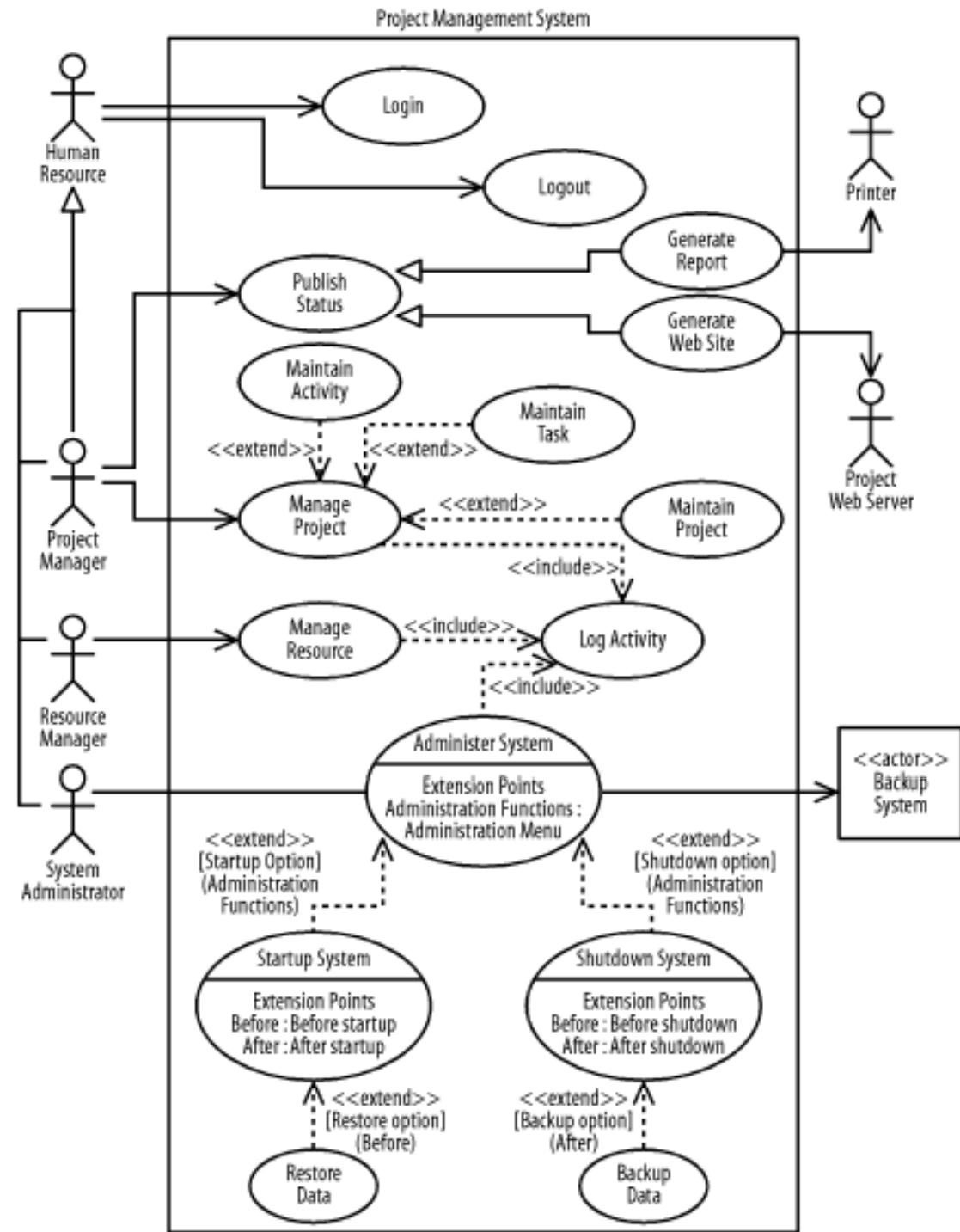
Ejemplo

Gestión de Pacientes



Ejemplo

Gestión de Proyectos



Casos de Uso

- Son útiles en tres áreas:
 - Especificación de requisitos
 - Comunicación con los clientes
 - Su simplicidad los convierte en excelentes medios de comunicación
 - Generación de casos de prueba
 - A partir de los escenarios de un Caso de Uso

Indice

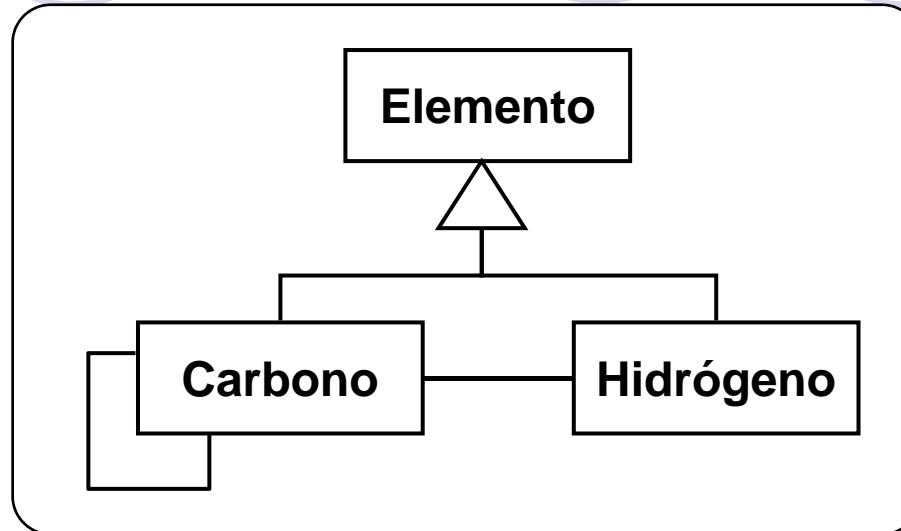
- Diagramas de Casos de Uso.
- **Diagramas de Estructura.**
 - **Clases y Objetos.**
 - **Componentes.**
 - **Estructuras Compuestas.**
 - **Despliegue.**
 - **Paquetes.**
- Diagramas de Comportamiento.
- OCL.
- Herramientas.
- Ejemplos.
- Bibliografía.

Clases y Objetos

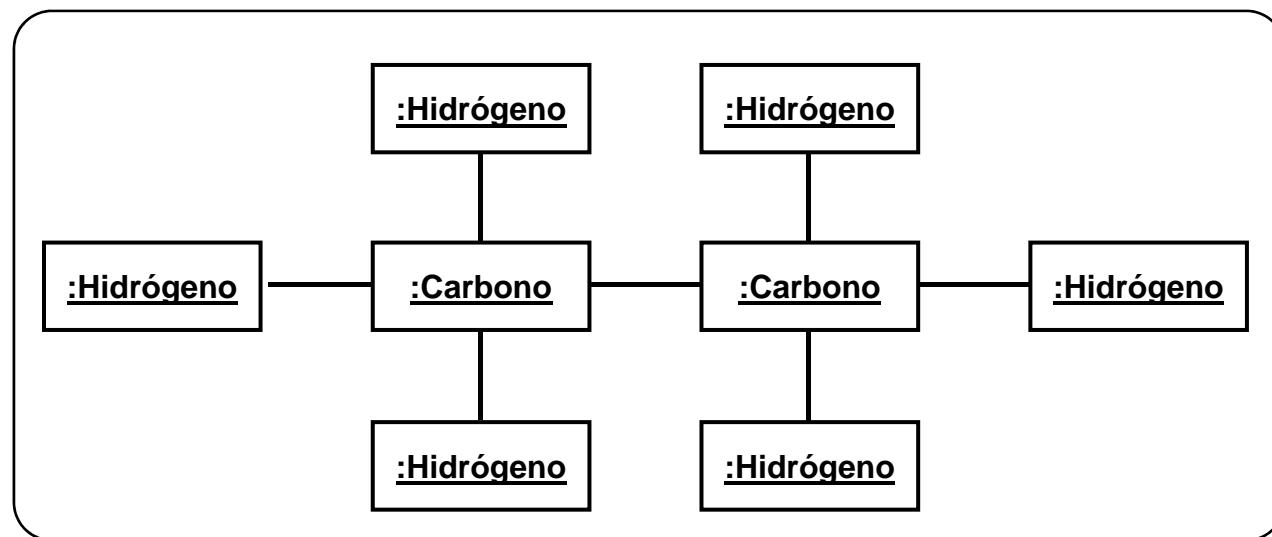
- Los diagramas de Clases y de Objetos son los principales modos de representar los aspectos estructurales en UML.
- ***Diagramas de clases.*** Estructura del sistema.
 - Clases.
 - Atributos: Tipos, valores iniciales.
 - Operaciones: visibilidad.
 - Relaciones con otras clases: Asociaciones
- ***Diagramas de objetos.*** Estructura del sistema en tiempo de ejecución.
 - Objetos. Instancias de una Clase.
 - Atributos (valores actuales).
 - *Links.* Relaciones entre objetos, instancias de asociaciones.

Clases y Objetos

*Diagrama de
clases*



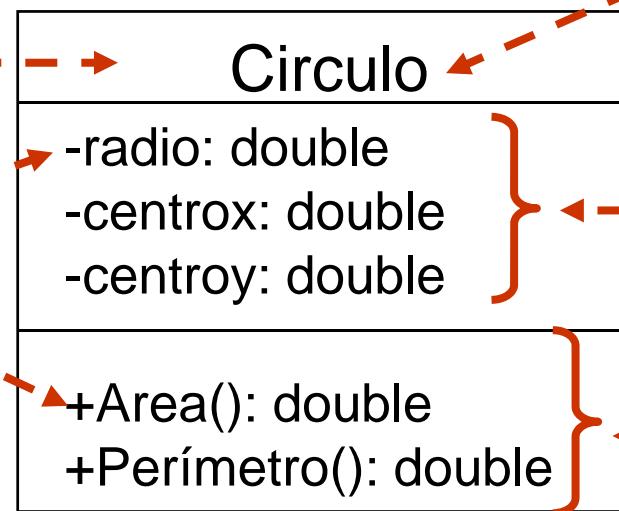
*Diagrama de
objetos*



Clases y Objetos

Nombre
de la clase

visibilidad

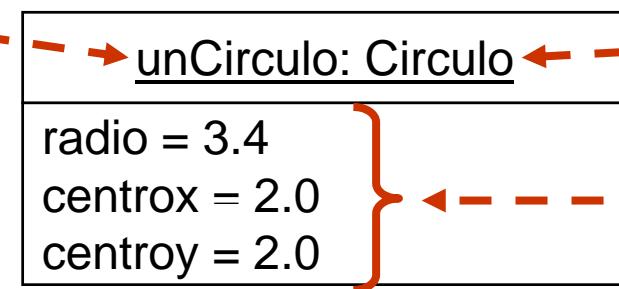


En cursiva si es
abstracta

Atributos

Operaciones

Nombre
del objeto



Clase
del objeto

Valores de
los atributos

Clases

Atributos

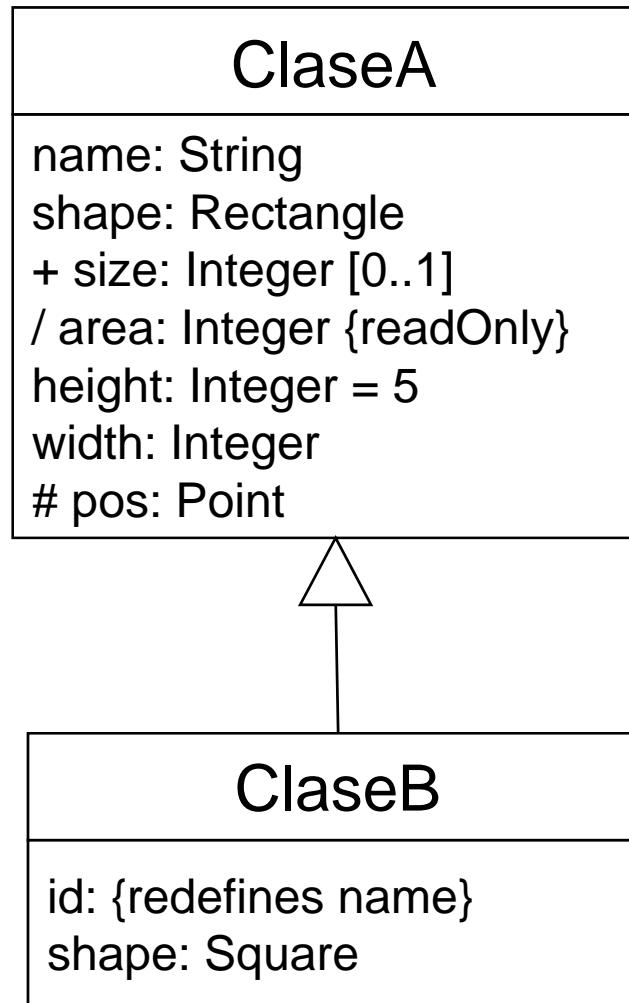
- Notación para atributos:

[*visibilidad*] [/] *nombre* [: *tipo*] [*multiplicidad*] [= *valor*] [{ *propiedad* }]

- Visibilidad (opcional):
 - Pública: +
 - Privada: -
 - Protegida: #
 - Paquete: ~
- "/" indica que el atributo es derivado.
- La multiplicidad va entre [] y por defecto vale 1.
- Propiedades válidas: {readOnly}, {union}, {subsets <property-name>}, {redefines <property-name>}, {ordered}, {bag}, {seq}, {sequence}, y {composite}.
- Un atributo subrayado es estático.

Clases

Ejemplo atributos



Clases

Métodos

- Notación para métodos:

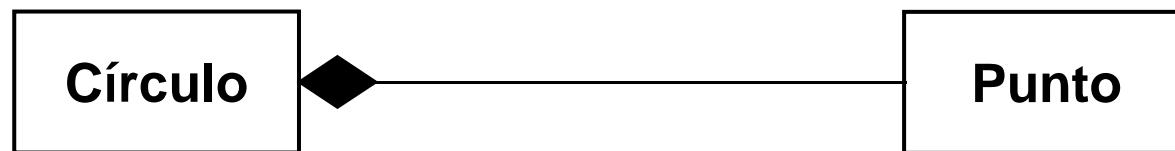
[visibilidad] nombre ([lista-parametros]) : [{propiedad}]

- Visibilidad (opcional).
- nombre del método
- lista de parámetros formales, separados por coma:
 dirección nombre : tipo [multiplicidad] = valor [{propiedad}]
- Los métodos estáticos se subrayan.
- Ejemplos:
 display ()
 -hide ()
 +createWindow (location: Coordinates, container: Container [0..1]): Window
 +toString (): String

Asociaciones

Composición

- Un *Círculo* contiene un *Punto*
- Se representa con una **Composición**



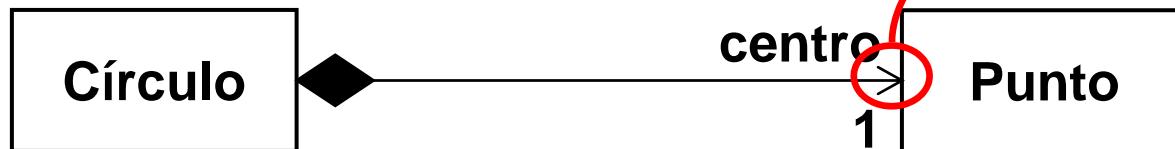
- Relación del tipo todo/parte
 - El todo es el *Círculo*
 - La parte es el *Punto*
- Es una relación **fuerte**
 - Si el *Círculo* es destruido o copiado, también lo es el *Punto*
 - La cardinalidad en la parte del todo es 0..1 o 1.

Asociaciones

Navegación, Roles, Cardinalidad

- Las asociaciones pueden tener etiquetas:

- Nombre
- Roles en la relación
- Multiplicidad (cardinalidad)



- Ejemplos de cardinalidad:

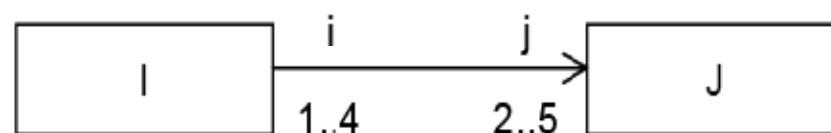
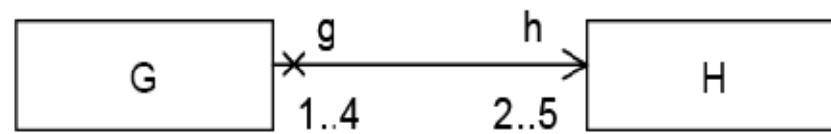
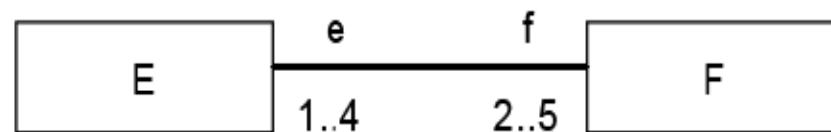
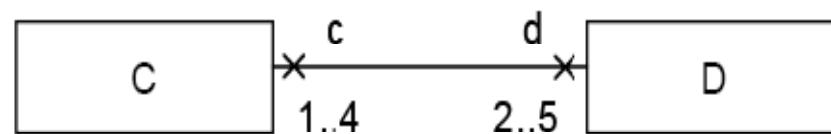
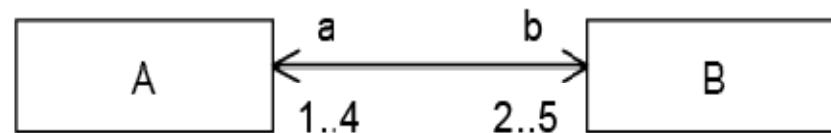
1..*	mínimo 1, no hay máximo
0..*	mínimo 0, no hay máximo
0..1	mínimo 0, máximo 1
1,2,4	uno, dos o cuatro
3	exactamente tres

- Navegación:

- Unidireccional
- Bidireccional
- No especificado.
- No navegable (x)

Asociaciones

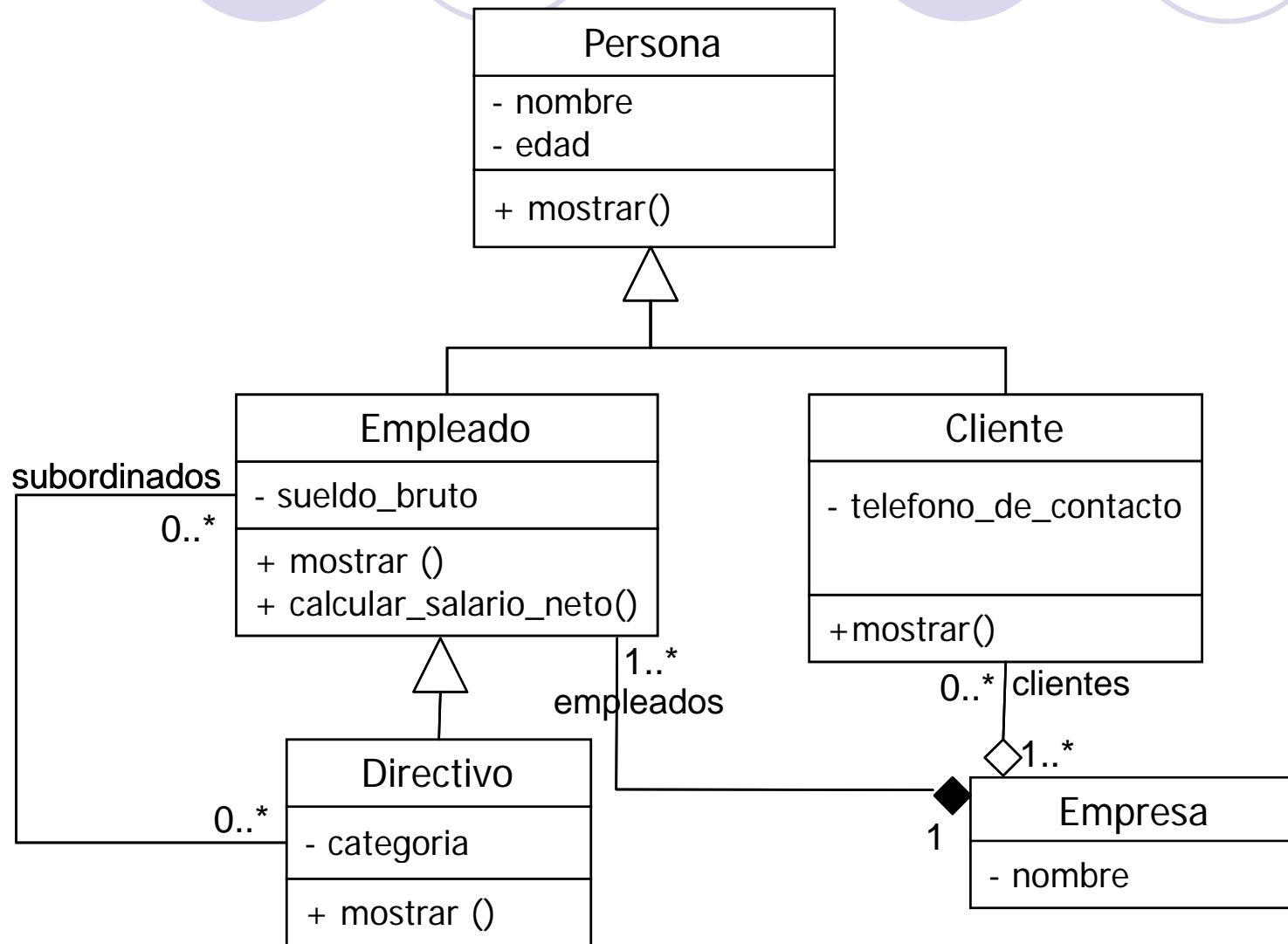
Ejemplos de Navegación y Cardinalidad



Ejercicio

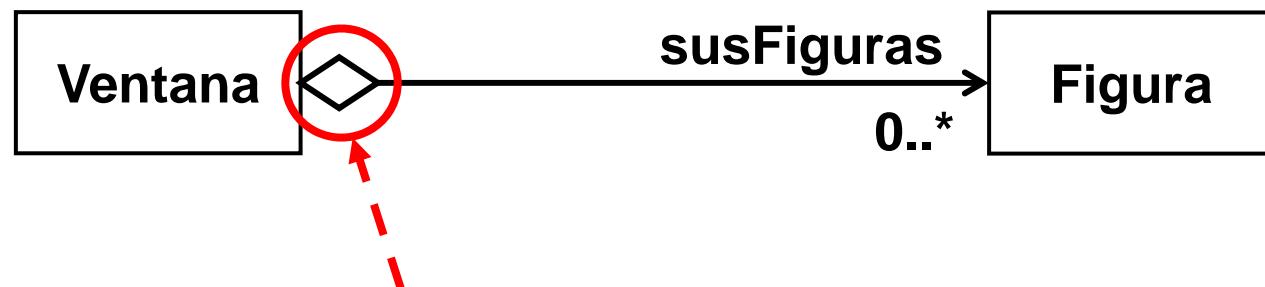
- Representa mediante un diagrama de clases la siguiente especificación:
 - Una aplicación necesita almacenar información sobre empresas, sus empleados y sus clientes.
 - Ambos se caracterizan por su nombre y edad.
 - Los empleados tienen un sueldo bruto, los empleados que son directivos tienen una categoría, así como un conjunto de empleados subordinados.
 - De los clientes además se necesita conocer su teléfono de contacto.
 - La aplicación necesita mostrar los datos de empleados y clientes.

Ejercicio



Asociaciones: Agregación

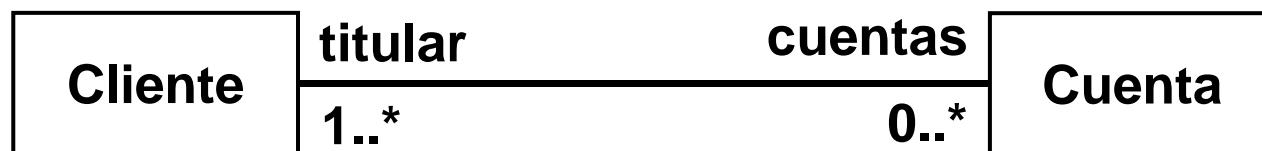
- Cuando la relación todo/parte no es tan fuerte, se utiliza **Agregación**



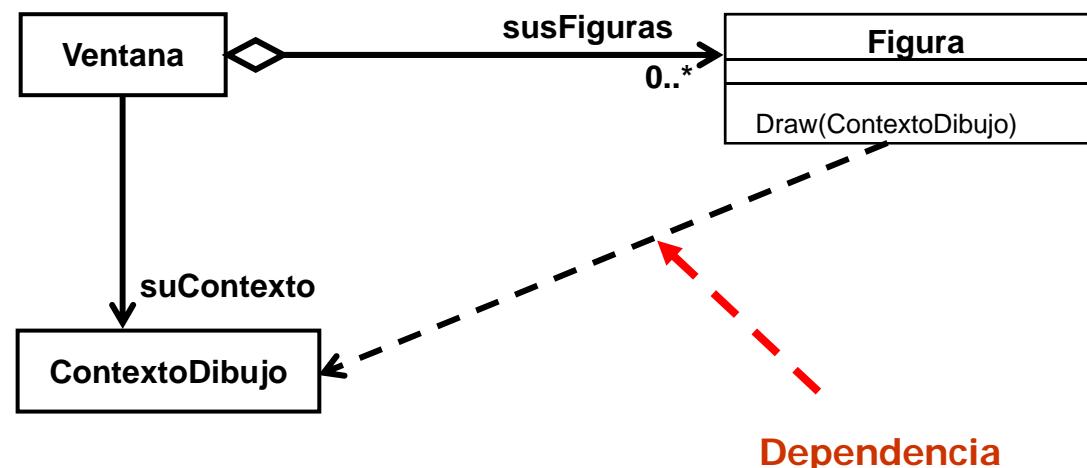
**La ventana contiene figuras,
pero cada una puede existir sin la otra**

Asociaciones y Dependencia.

- Existen relaciones de conocimiento entre clases que no implican una relación todo/parte

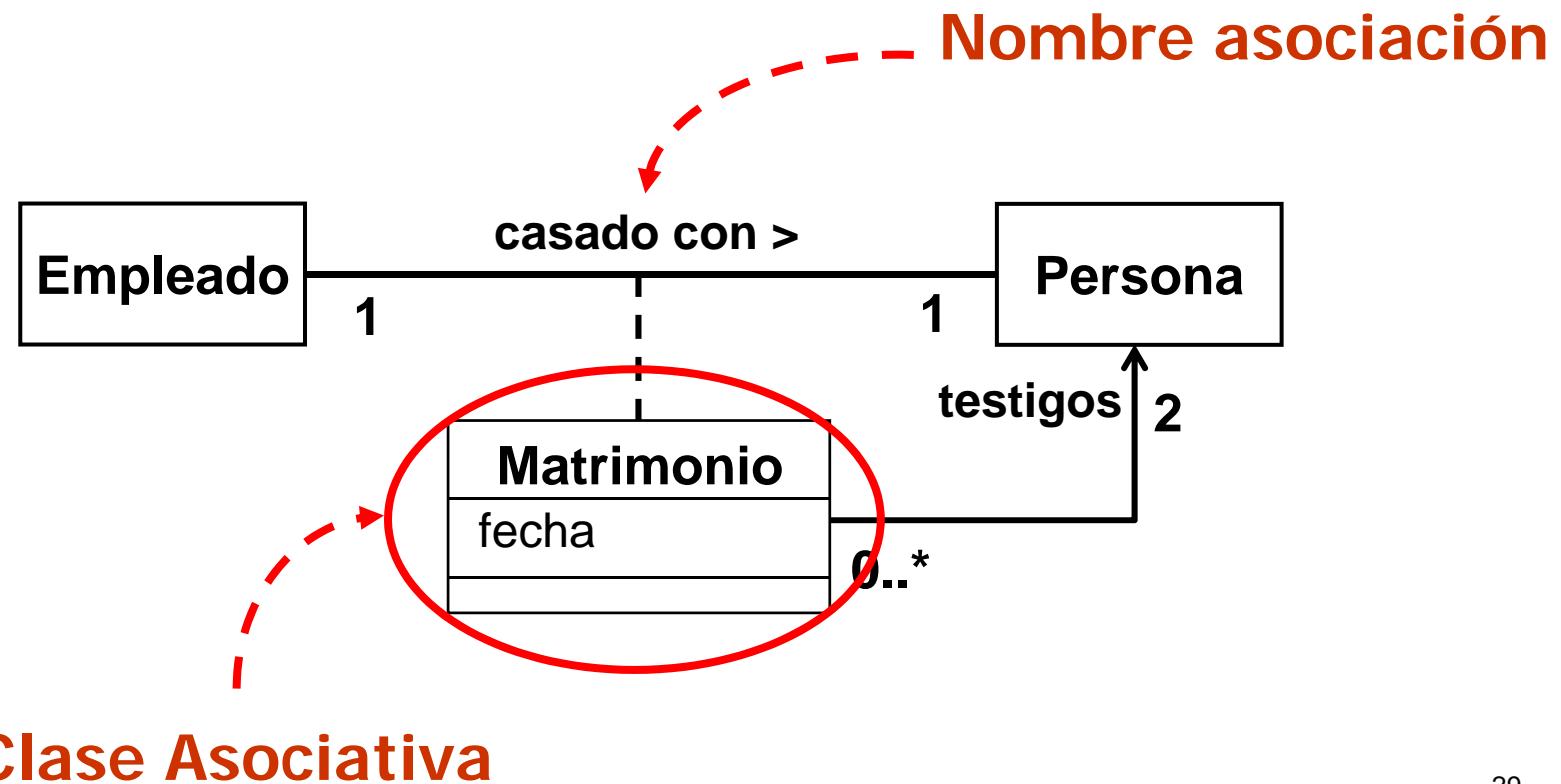


- Dependencia: relación muy débil.



Clases Asociativas

- Asociación con atributos propios.

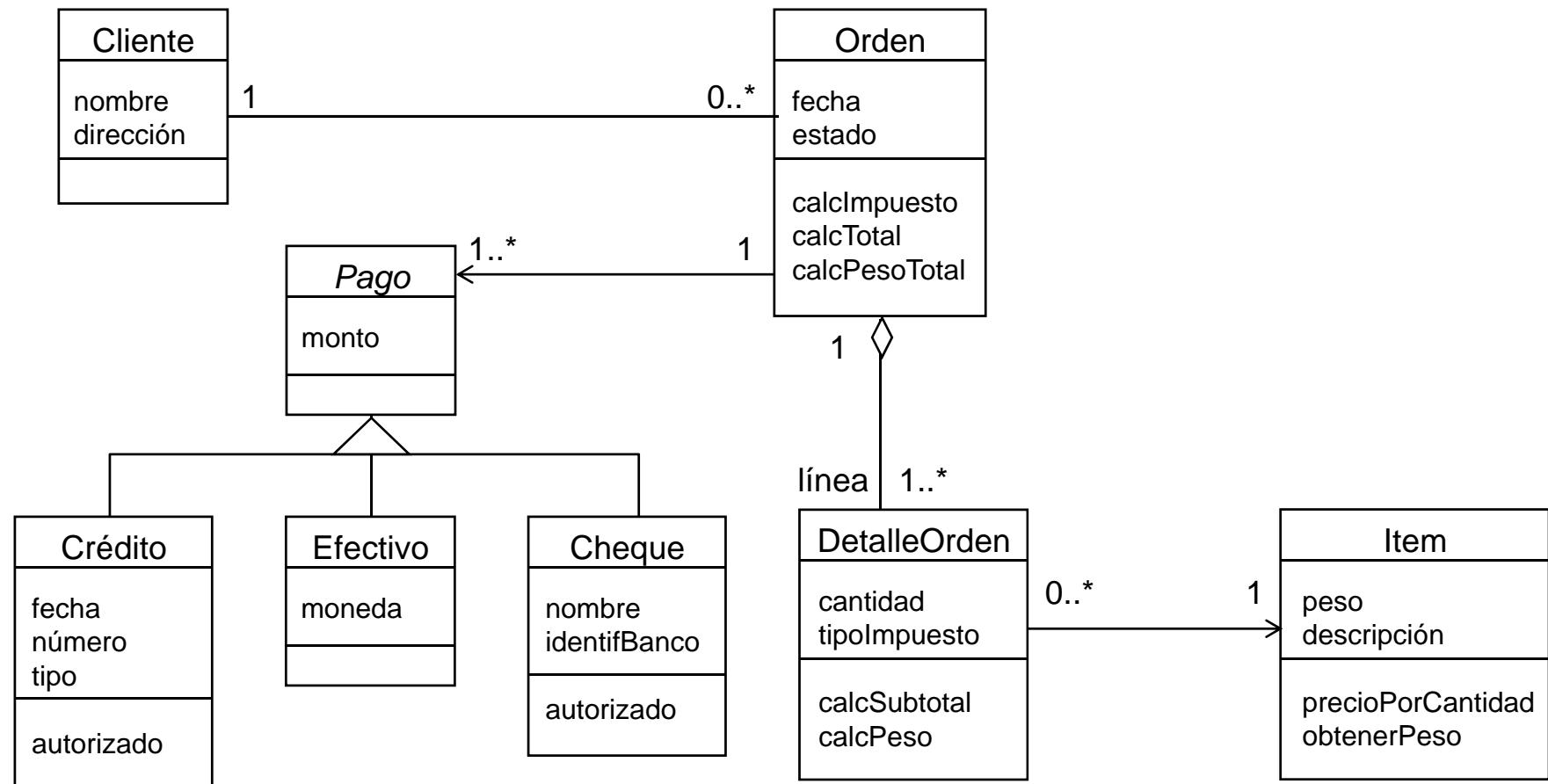


Clases y Objetos

Estilo

- Los atributos no deben ser objetos (utilizar relaciones en tal caso).
- En los diagramas de clases no suelen aparecer (son detalles de implementación y no de diseño):
 - Constructores
 - Métodos de acceso (“get/set”)
 - Métodos de gestión de elementos de una asociación o agregación (por ejemplo, “add/remove”)

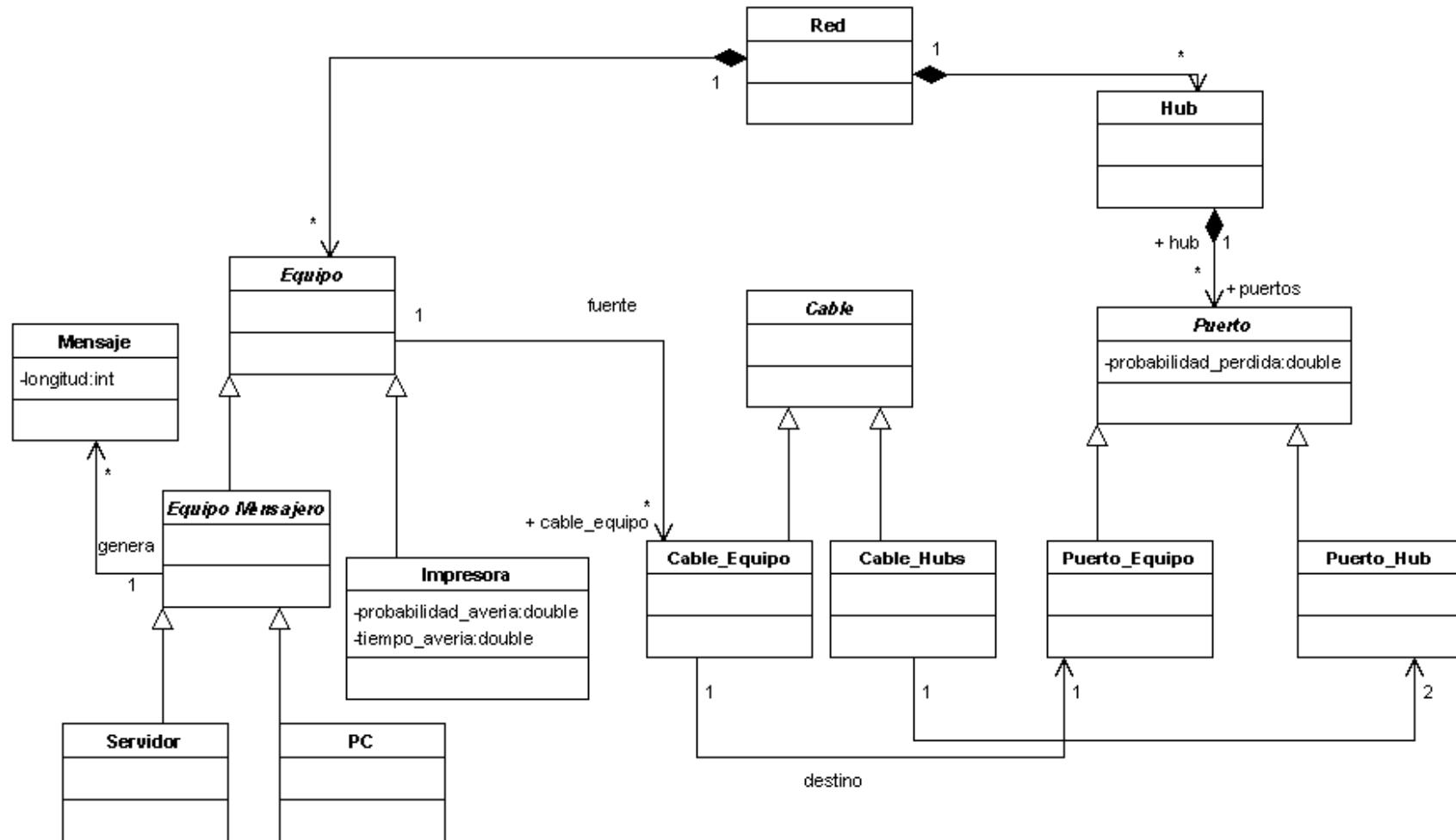
Ejemplo



Ejercicio

- Especificar un diagrama de clases que describa redes de ordenadores.
- Los elementos que se pueden incluir en la red son:
 - Servidor, PC, Impresora.
 - Hub, Cable de red.
- Los PCs pueden conectarse con un único Hub, los servidores con uno o varios.
- Los Servidores y PCs pueden generar mensajes, con una cierta longitud.
- Los Hubs tienen un número de puertos, algunos de los cuales puede usarse para conectar con otros Hubs. Tienen cierta probabilidad de “perder” mensajes.
- Las impresoras pueden averiarse, con cierta probabilidad, durante cierto tiempo.

Ejercicio. Posible Solución.

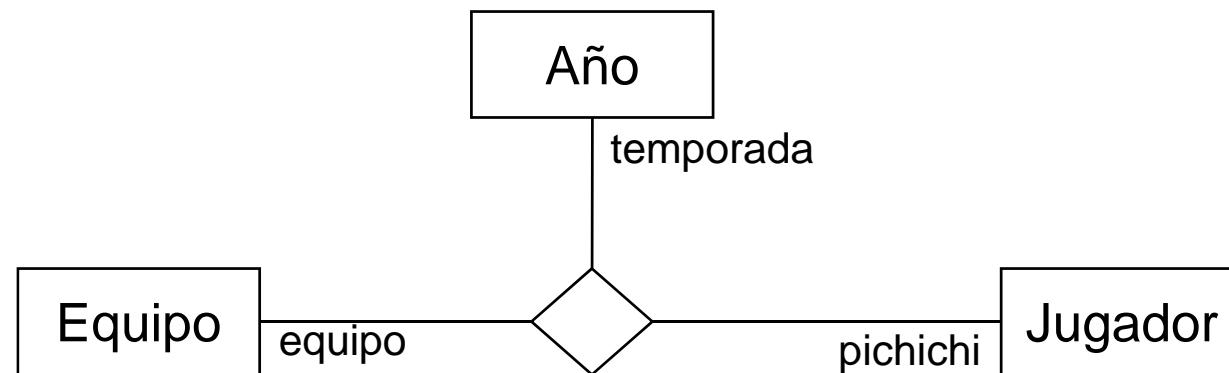


“Los PCs pueden conectarse con un único Hub, los servidores con uno o varios”
 Podemos modelarlo como una restricción OCL, o bien añadir asociaciones desde Servidor y PC

Más sobre asociaciones

Asociaciones n-arias

- Asociaciones entre más de dos clases:



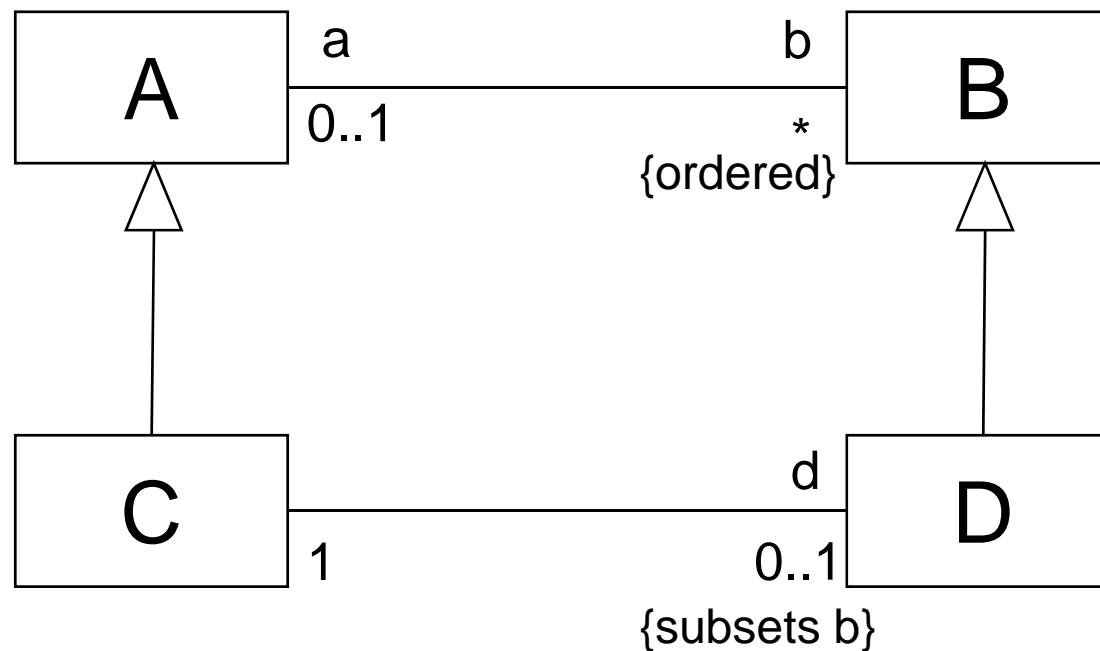
Más sobre asociaciones

Adornos en asociaciones y fin de asociación.

- Asociaciones derivadas (con un “/” delante del nombre).
- Propiedades, cerca del nombre de la asociación.
- Los finales de la asociación pueden adornarse con:
 - Multiplicidad.
 - Nombre (rol).
 - Propiedades:
 - {subsets <nombre-prop>}.
 - {redefine <nombre-fin-asoc>}.
 - {union}.
 - {ordered} (un conjunto ordenado).
 - {bag} (conjunto con repetición).
 - {sequence} o {seq} (bag ordenado).

Más sobre asociaciones

Adornos en asociaciones y fin de asociación: Ejemplos

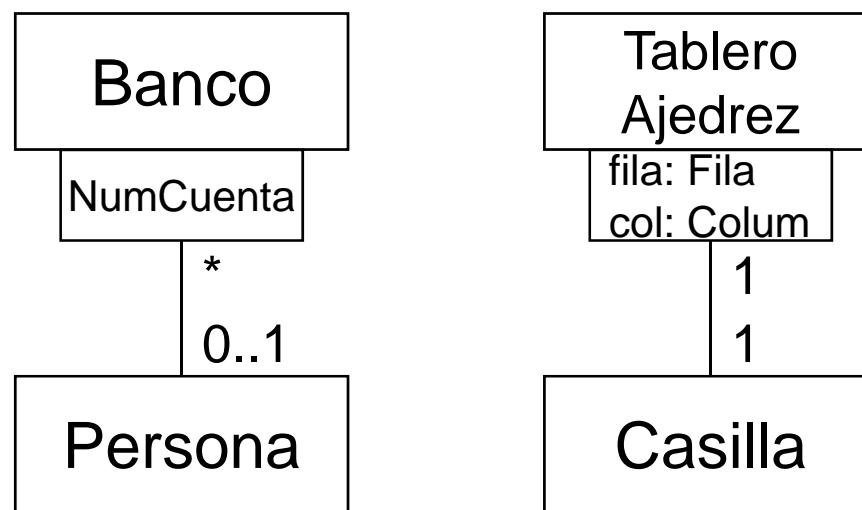


- Para un objeto de tipo C, la colección d es un subconjunto de la colección b.

Más sobre asociaciones

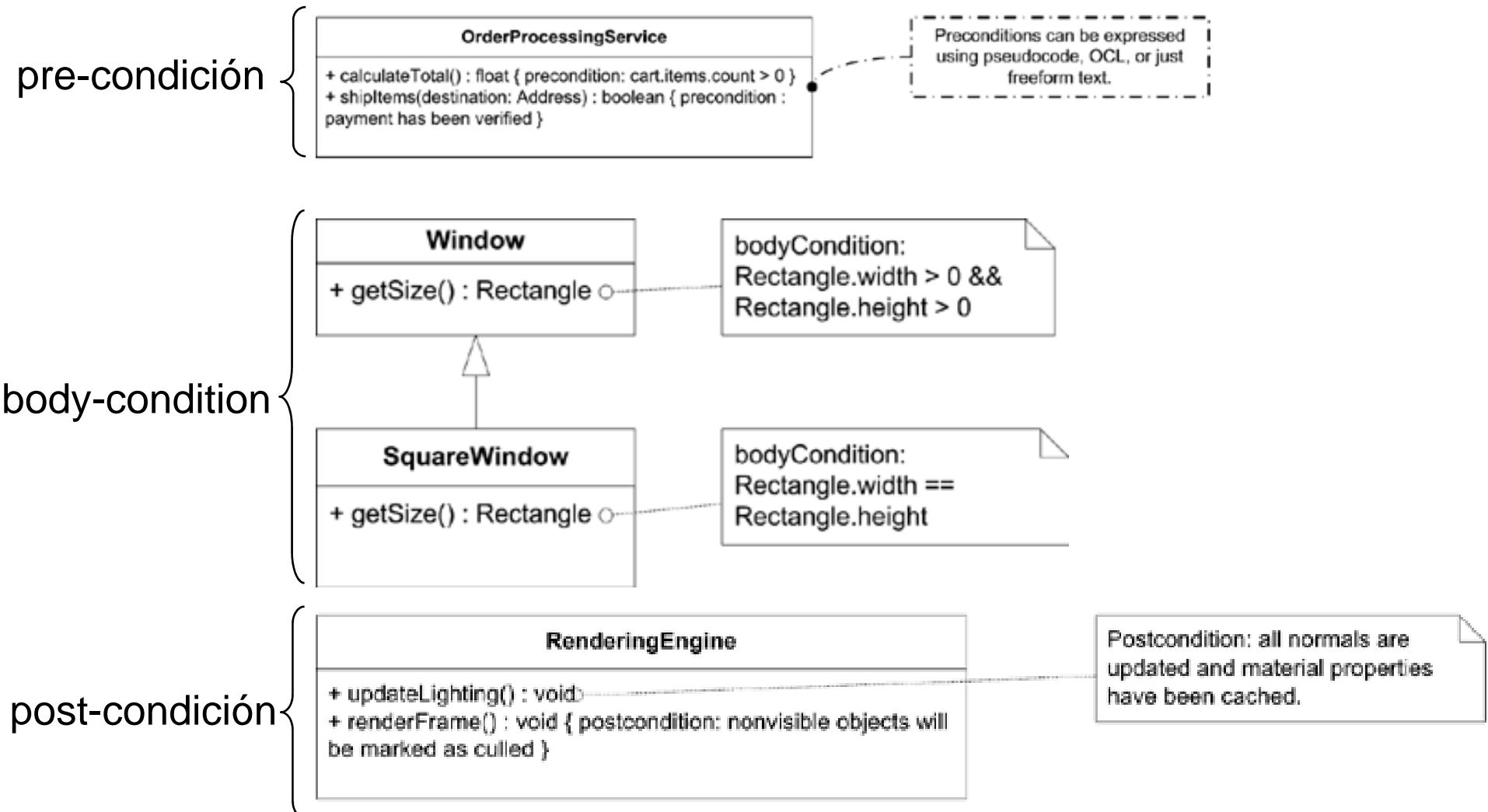
Asociaciones Cualificadas

- Un cualificador declara una partición del conjunto de instancias asociadas con respecto a la instancia cualificada.

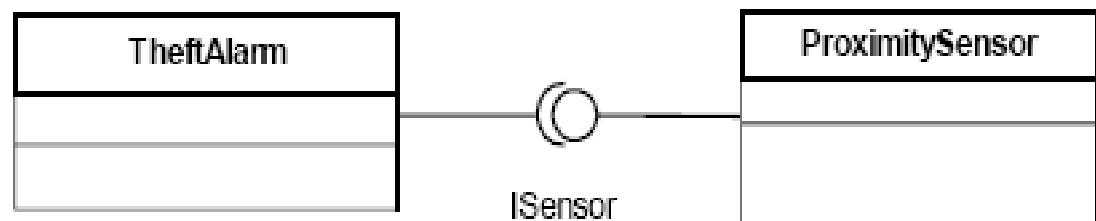
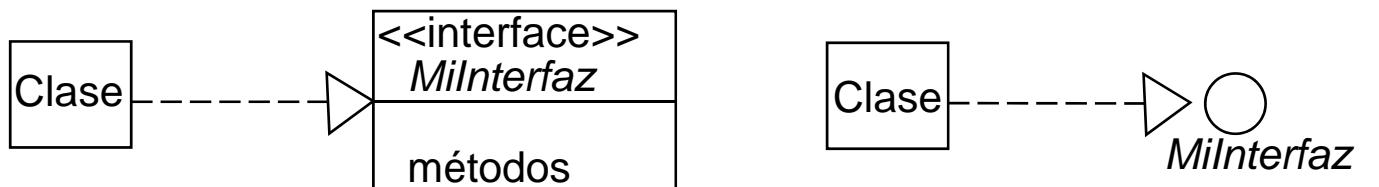


- Dado un objeto cualificado, el número de objetos al otro lado de la asociación viene dado por la multiplicidad declarada.
 - 0..1 : el valor del cualificador es único.
 - 0..* : el conjunto de instancias asociadas se partitiona en subconjuntos.
- Similar a un array asociativo, map o tabla hash.

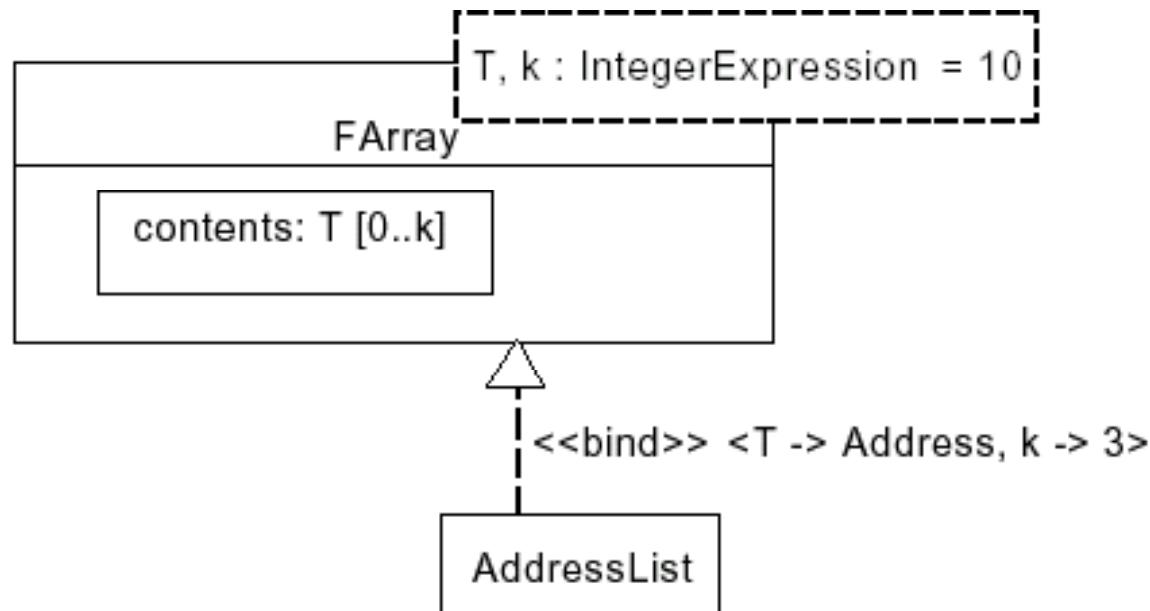
Pre- y Post- Condiciones, Notas



Interfaces



Plantillas



`FArray<T -> Point>`

Una clase anónima ligada:

○ $T \rightarrow \text{Point}$

○ $K \rightarrow 10$

Ejercicio

Examen Junio 2008.

Realiza el diseño de una aplicación para la gestión de pedidos. La aplicación deberá manejar clientes (se guarda su nombre, dirección, teléfono y e-mail), que pueden realizar pedidos de productos, de los cuales se anota la cantidad en stock. Un cliente puede tener una o varias cuentas para el pago de los pedidos. Cada cuenta está asociada a una tarjeta de crédito, y tiene una cierta cantidad disponible de dinero, que el cliente debe aumentar periódicamente para poder realizar nuevos pedidos.

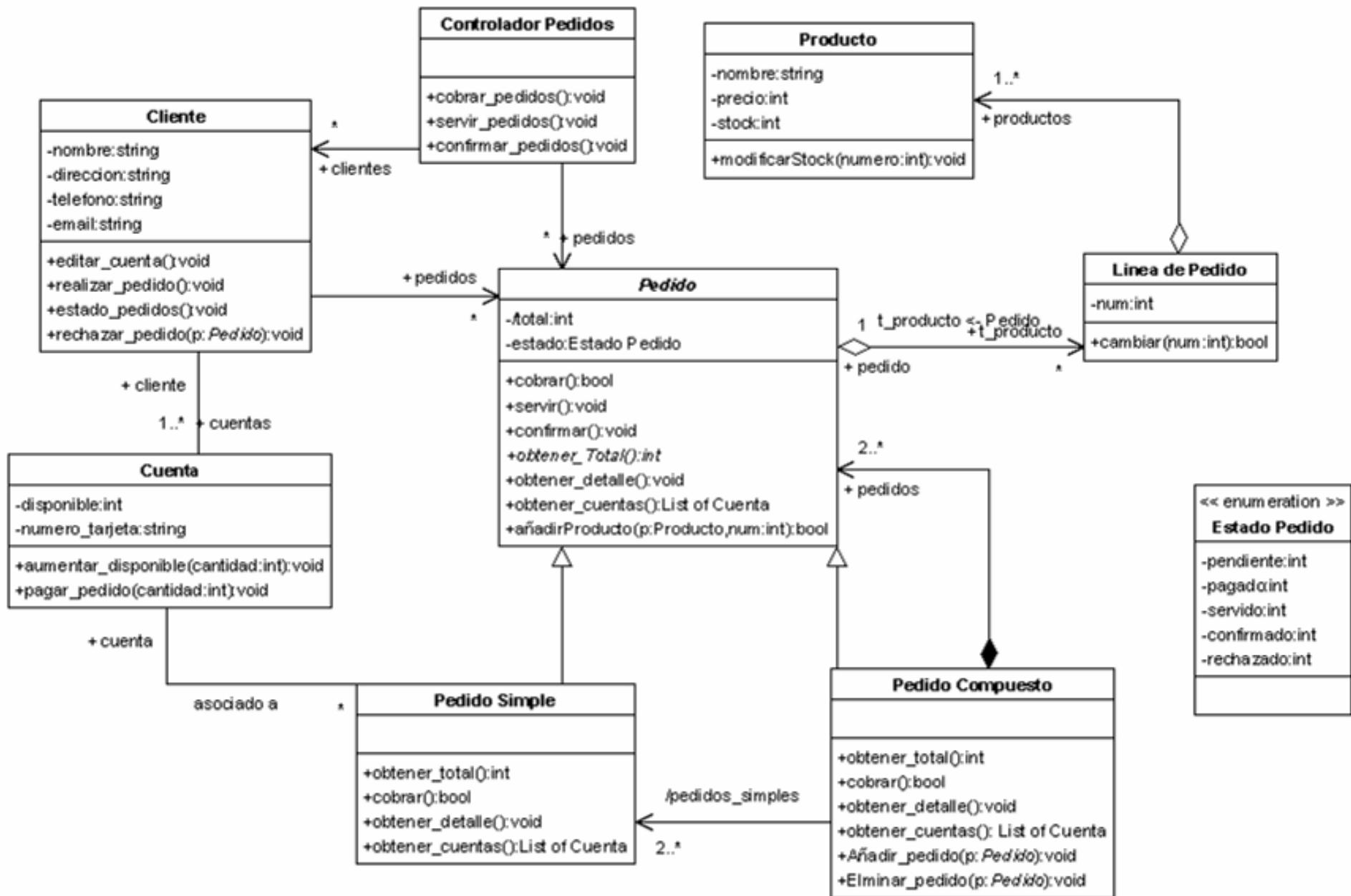
Un cliente puede empezar a realizar un pedido sólo si tiene alguna cuenta con dinero disponible. Al realizar un pedido, un cliente puede agruparlos en pedidos simples o compuestos. Los pedidos simples están asociados a una sola cuenta de pago y (por restricciones en la distribución) contienen un máximo de 20 unidades del mismo o distinto tipo de producto. A su vez, un pedido compuesto contiene dos o más pedidos, que pueden ser simples o compuestos. Como es de esperar, el sistema debe garantizar que todos los pedidos simples que componen un pedido compuesto se paguen con cuentas del mismo cliente. Además, sólo es posible realizar peticiones de productos en stock.

Existe una clase (de la cual debe haber una única instancia en la aplicación) responsable del cobro, orden de distribución y confirmación de los pedidos. El cobro de los pedidos se hace una vez al día, y el proceso consiste en comprobar todos los pedidos pendientes de cobro, y cobrarlos de la cuenta de pago correspondiente. Si una cuenta no tiene suficiente dinero, el pedido se rechaza (si es parte de un pedido compuesto, se rechaza el pedido entero). Una vez que el pedido está listo para servirse, se ordena su distribución, y una vez entregado, pasa a estar confirmado.

41

Se pide un diagrama de clases de diseño.

Solución



Solucion (ii)

- Nota: pedidos_simples es una asociación derivada. El atributo total de Pedido es derivado.
- Habría que incluir las siguientes restricciones OCL:

Context realizar_pedido:

pre: self.cuentas->exists(c | c.disponible > 0)

Context Pedido Compuesto:

inv: self.pedidos_simples->cuenta->cliente->asSet()->size() = 1

Context Pedido:

inv: self.t_productos.num->sum() <= 20

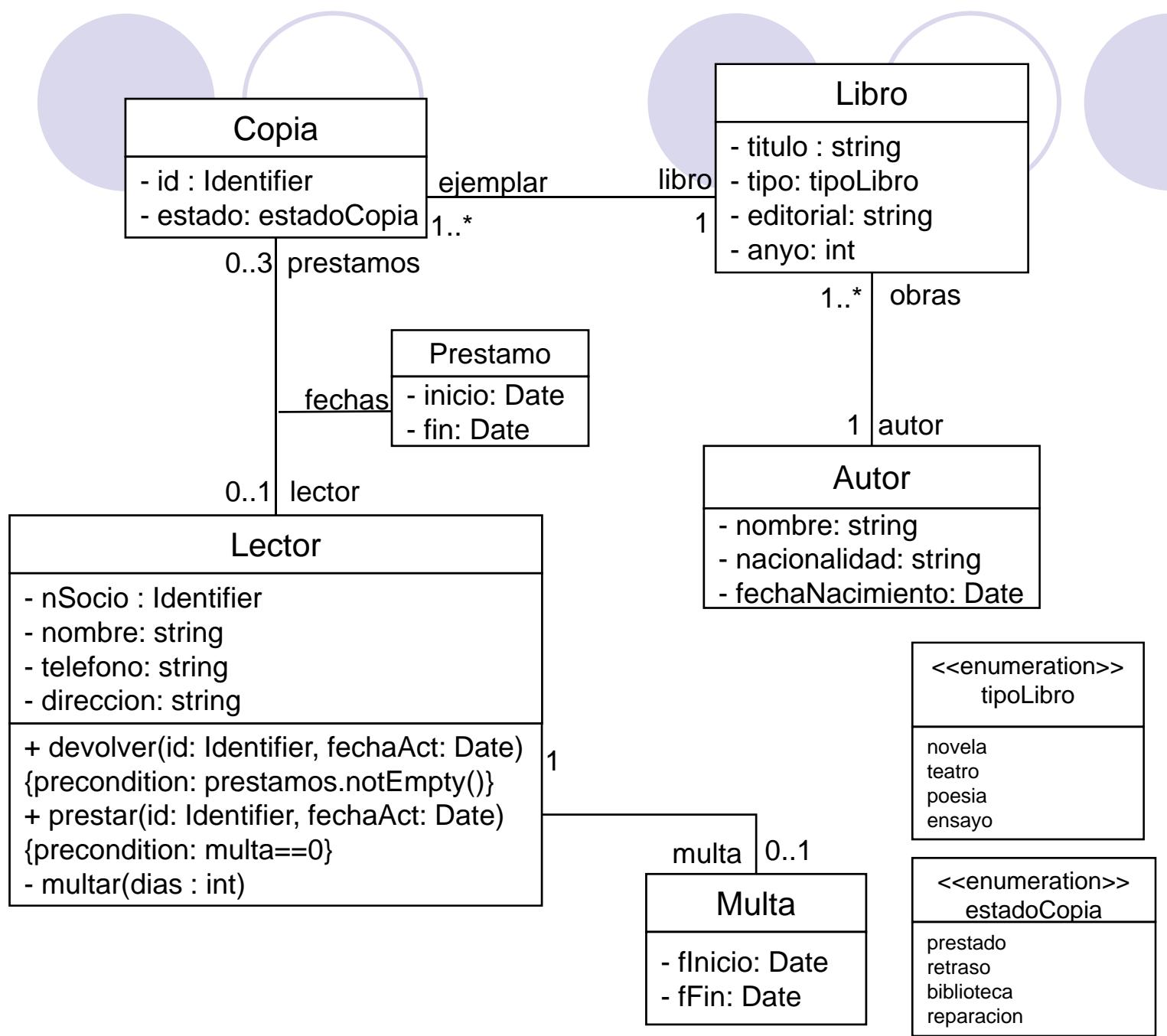
Context añadir_pedido(p: Producto, num: int):

pre: p.stock>=num

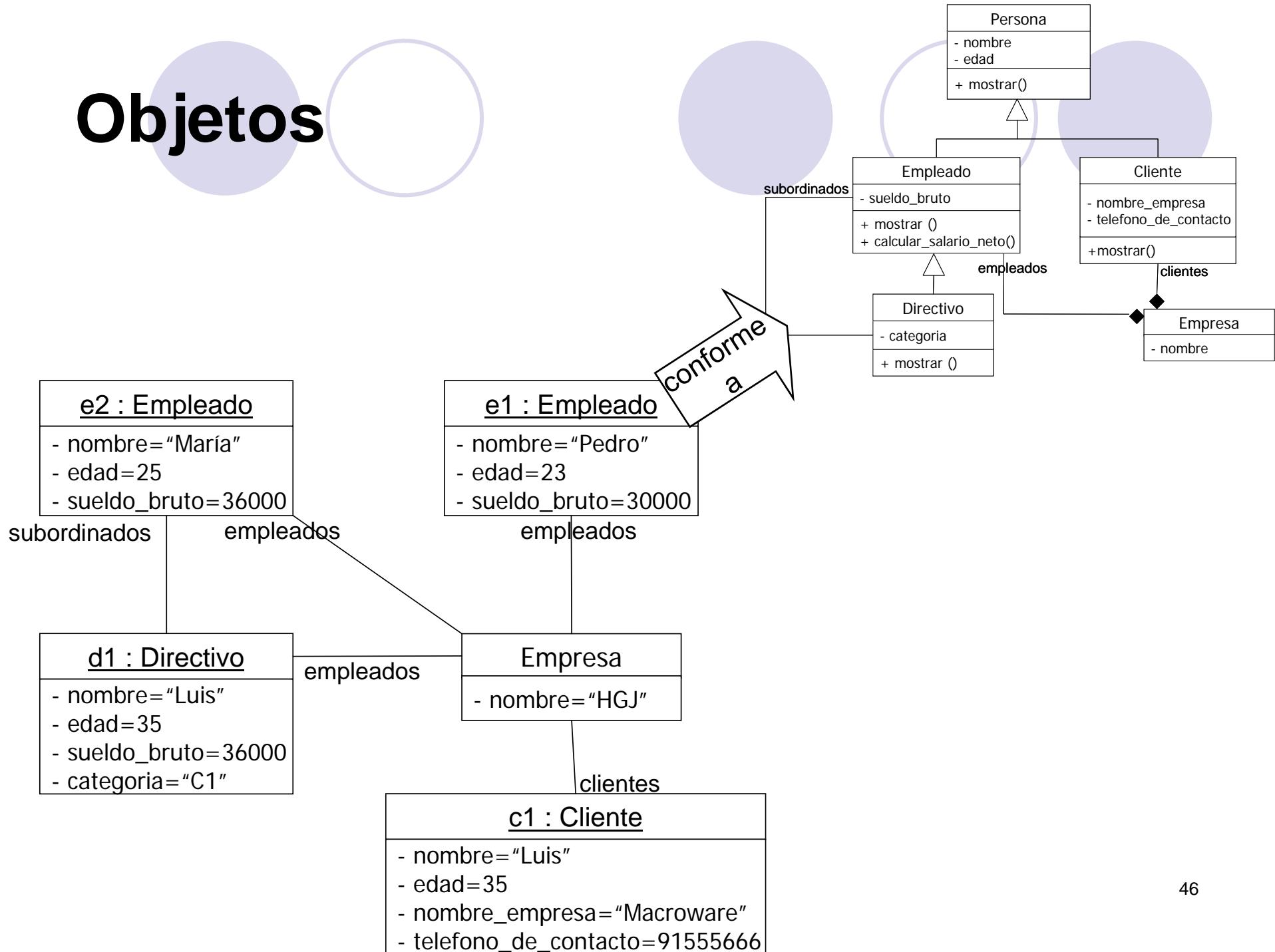
- Se usa el composite y el singleton (para la clase “Controlador Pedidos”, aunque esto no queda reflejado en el diseño a este nivel de abstracción).

Ejercicio: Biblioteca

- Una biblioteca tiene copias de libros. Estos últimos se caracterizan por su nombre, tipo (novela, teatro, poesía, ensayo), editorial, año y autor.
- Los autores se caracterizan por su nombre, nacionalidad y fecha de nacimiento.
- Cada copia tiene un identificador, y puede estar en la biblioteca, prestada, con retraso o en reparación.
- Los lectores pueden tener un máximo de 3 libros en préstamo.
- Cada libro se presta un máximo de 30 días, por cada día de retraso, se impone una “multa” de dos días sin posibilidad de coger un nuevo libro.
- Realiza un diagrama de clases y añade los métodos necesarios para realizar el préstamo y devolución de libros.



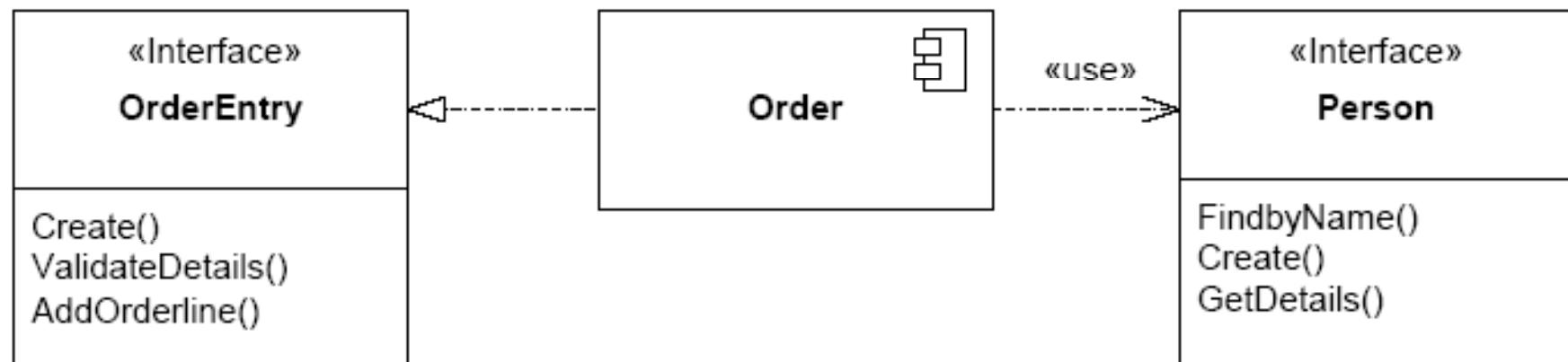
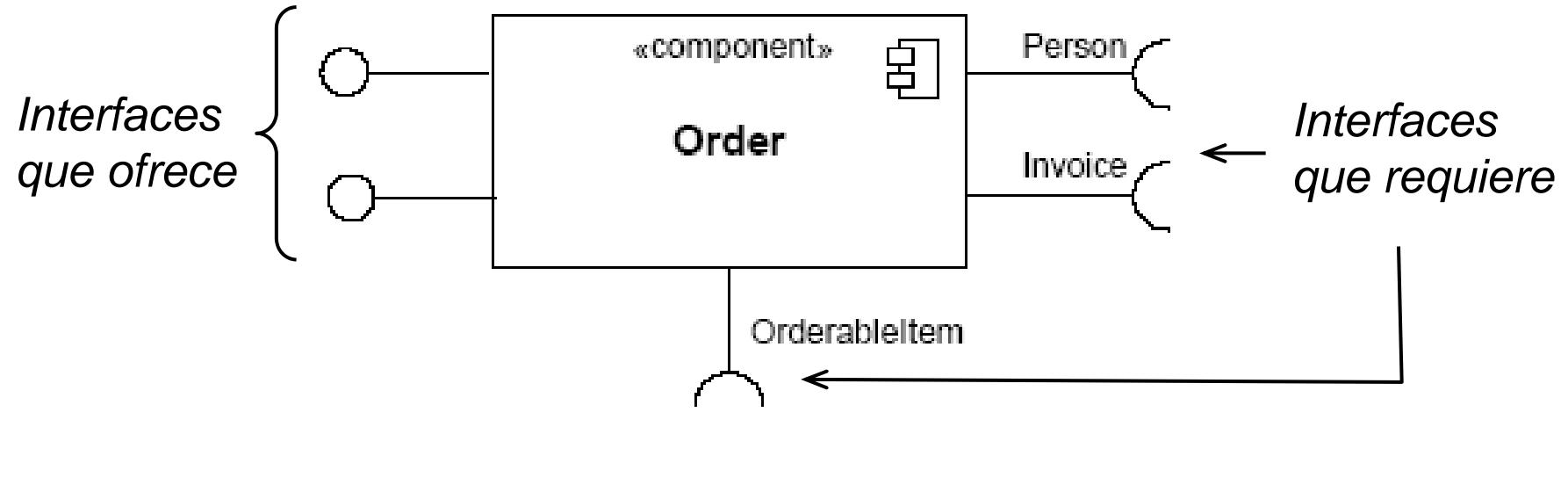
Objetos



Componentes

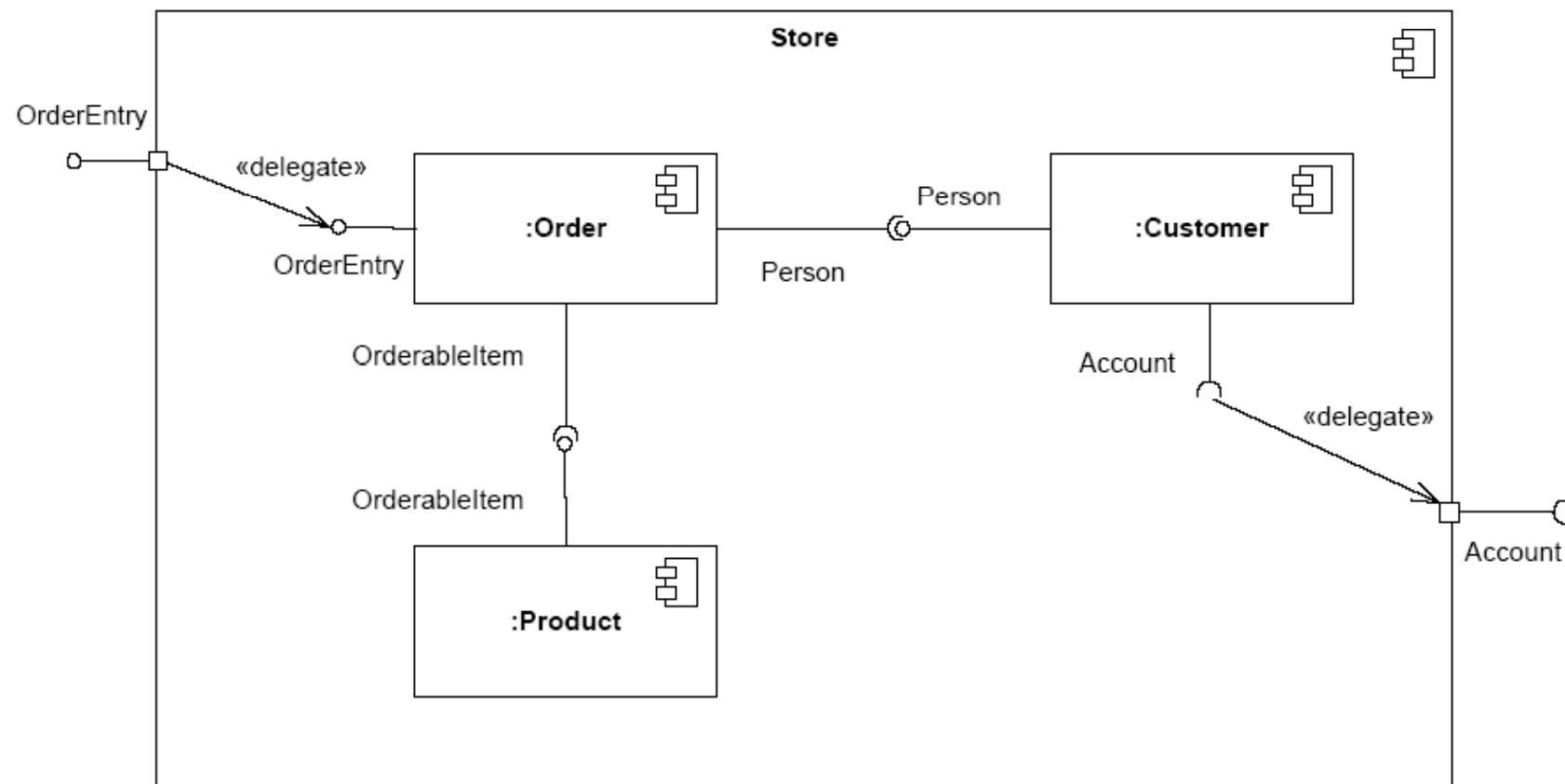
- Componente = “*Unidad Modular con interfaces bien definidos que es reemplazable en su entorno*”.
- Énfasis en reutilización y encapsulamiento.
- Servicios que provee y requiere (interfaces).
- Componentes lógicos (componentes de negocio, de proceso) y físicos (EJB, CORBA, COM+, .NET, ...)

Componentes



Componentes

Vista de caja blanca

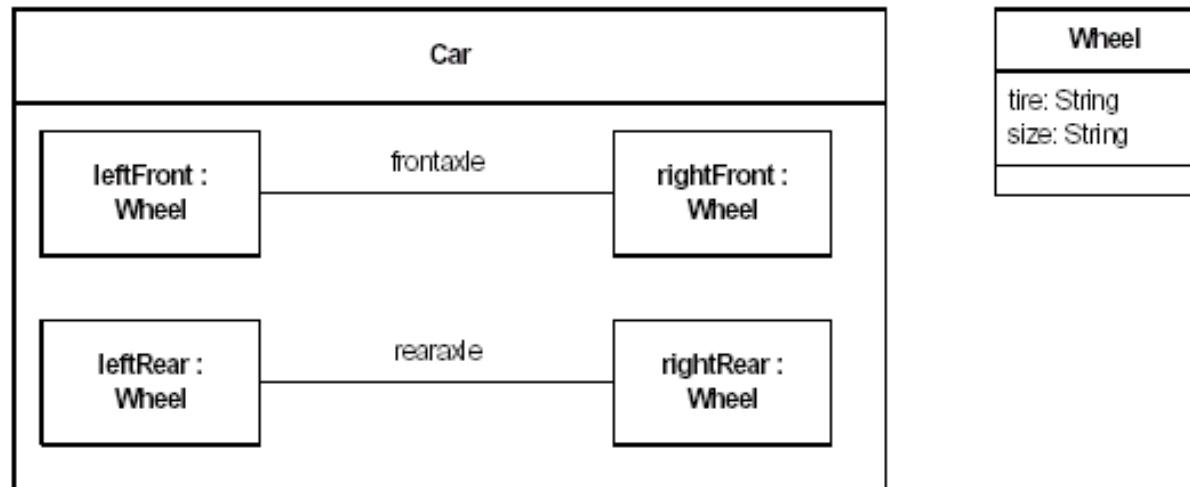


Estructuras Compuestas

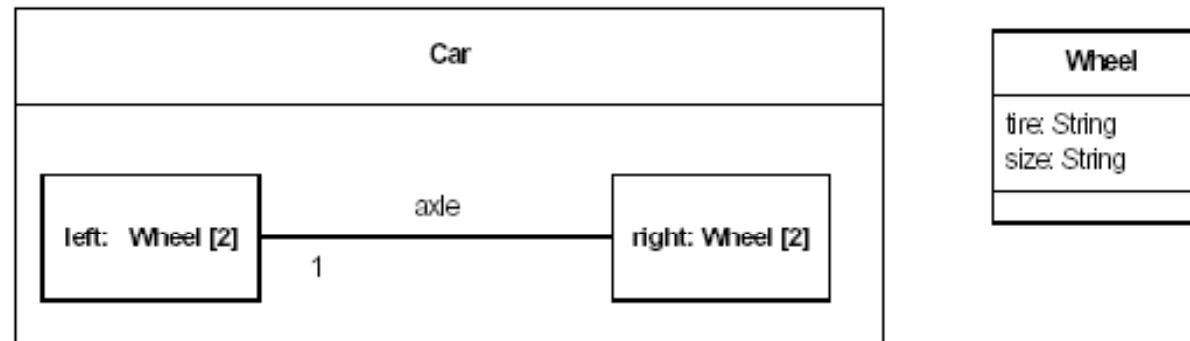
- Composición de elementos (clasificadores o colaboraciones)
- Instancias en tiempo de ejecución que colaboran a través de enlaces para alcanzar objetivos comunes.
- Colaboraciones: a través de roles.
- Se añade a las clases estructura internas y puertos.

Estructuras Compuestas

Estructura interna de una clase.

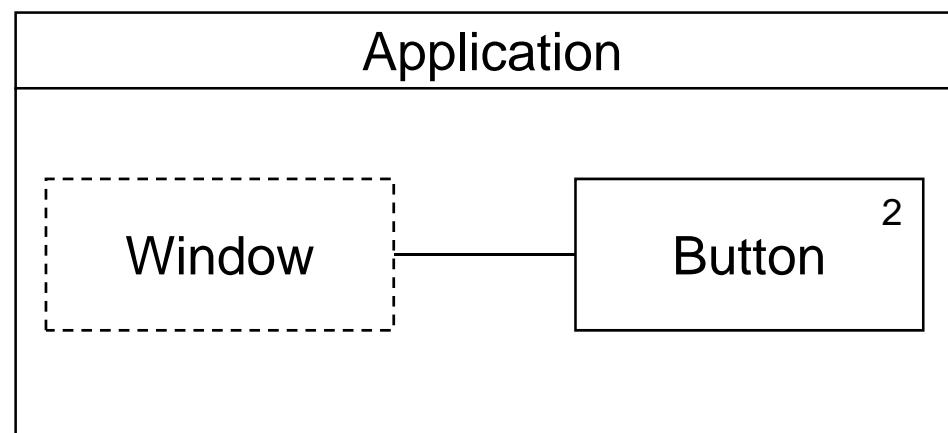
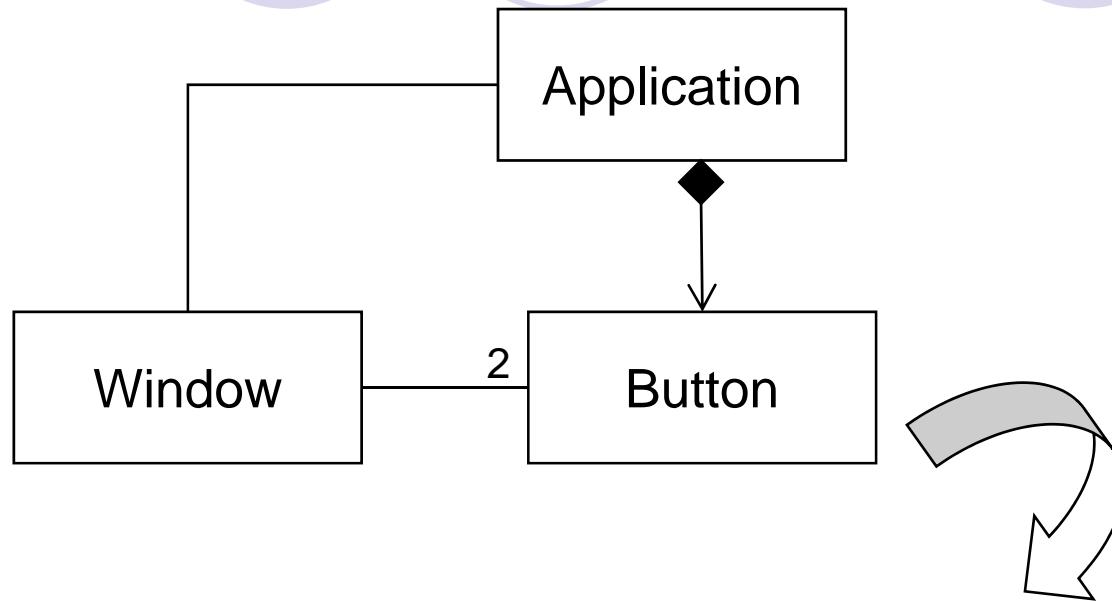


Estructura interna de una clase. Multiplicidades.



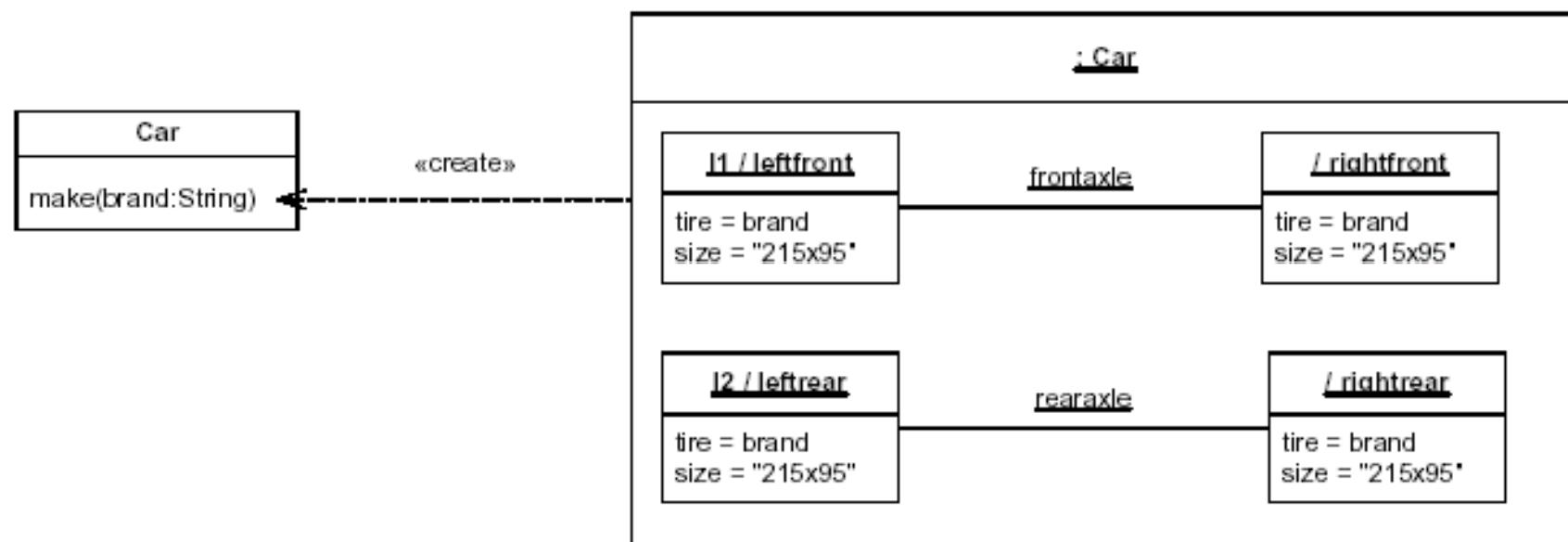
Estructuras Compuestas

Alternativa a relaciones de composición



Estructuras Compuestas

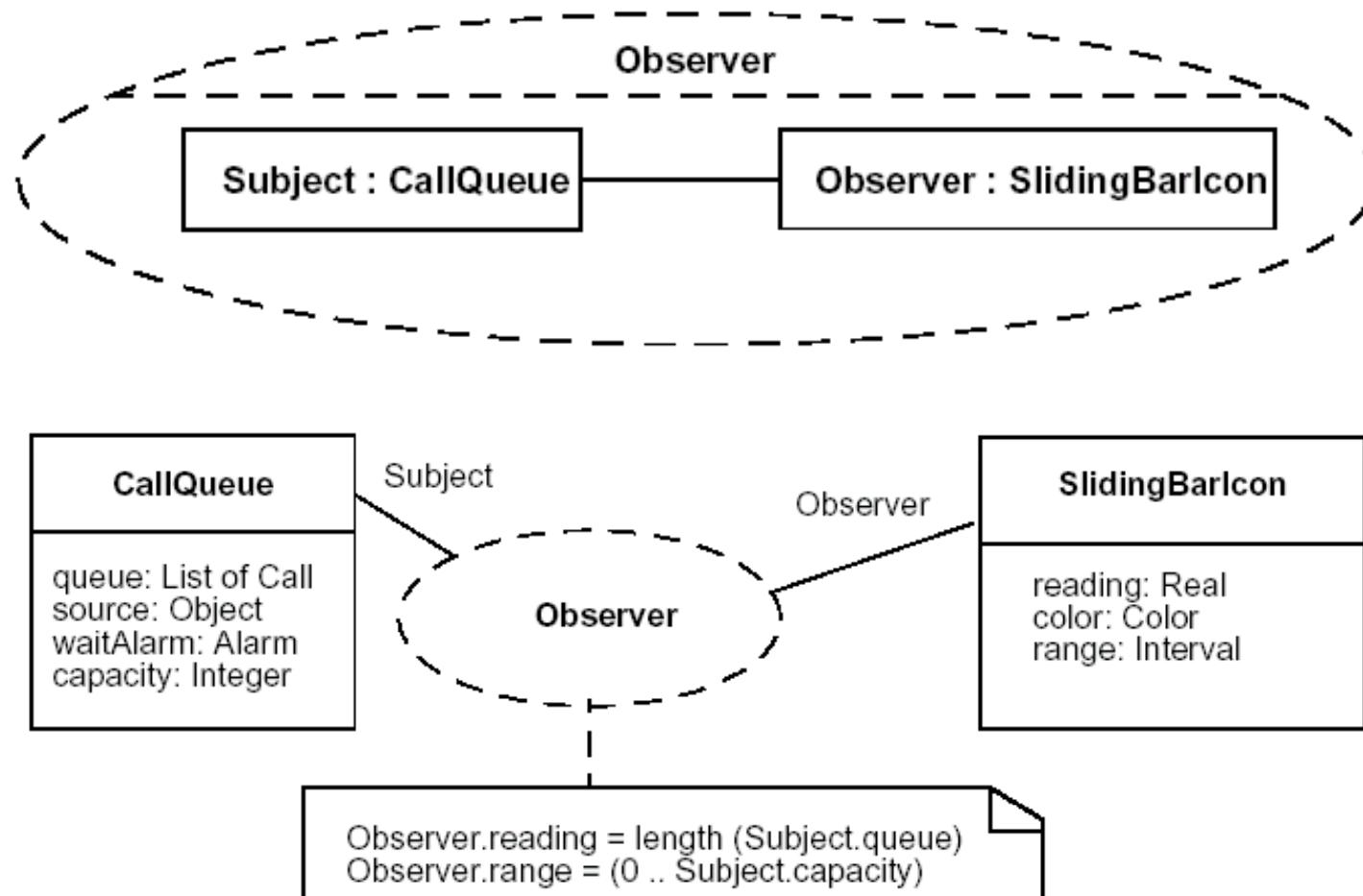
Estructura interna de una clase. Constructor.



Estructuras Compuestas

Colaboración.

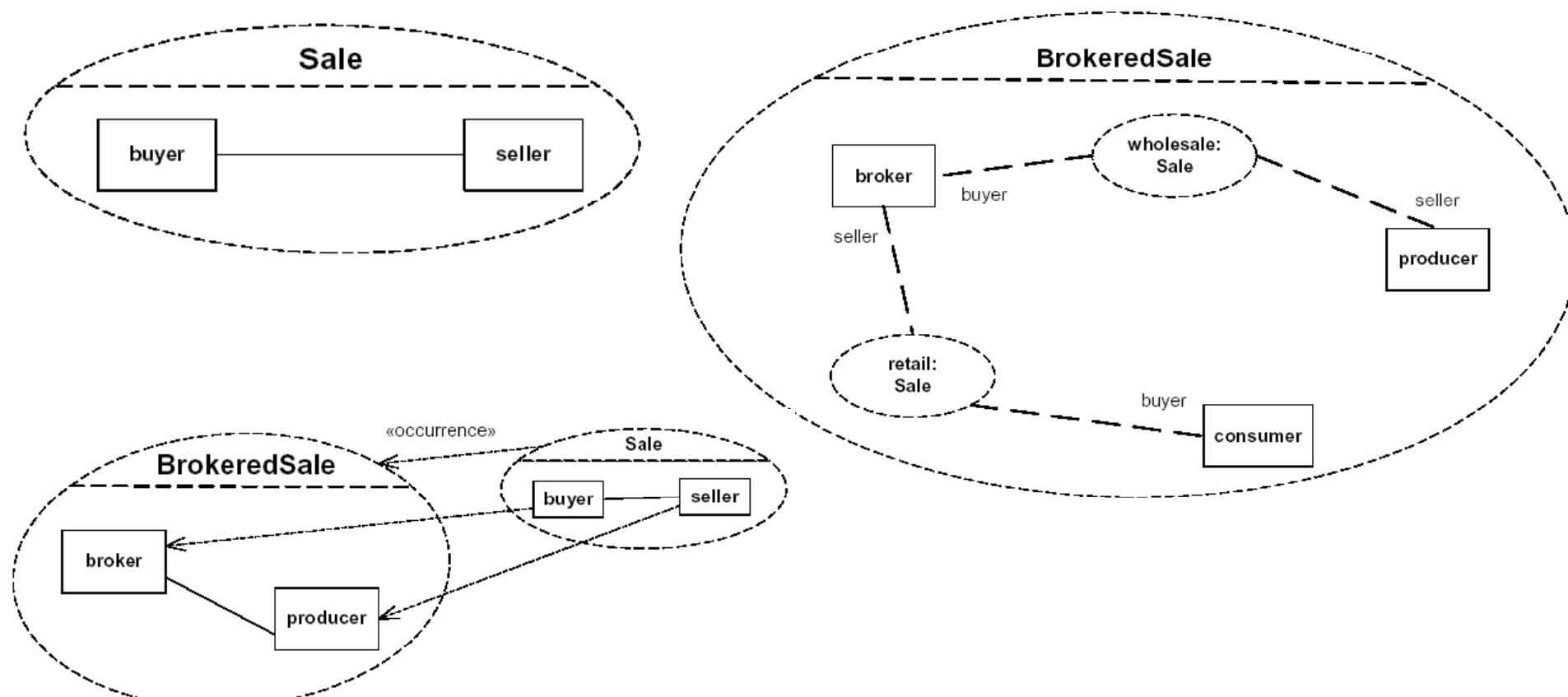
Estructura interna de una colaboración.



Estructuras Compuestas

Colaboración.

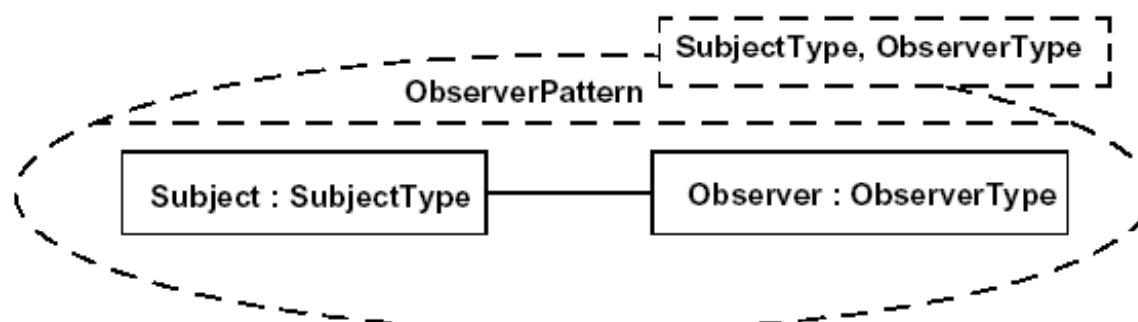
Ocurrencia de una colaboración en otra. Binding.



Estructuras Compuestas

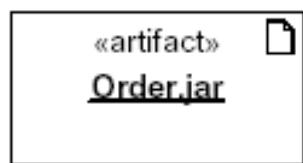
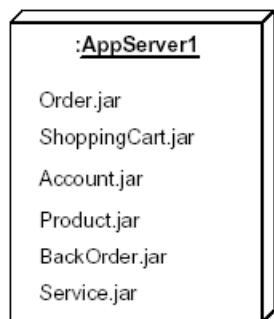
Colaboración.

Template de colaboración.



Despliegue

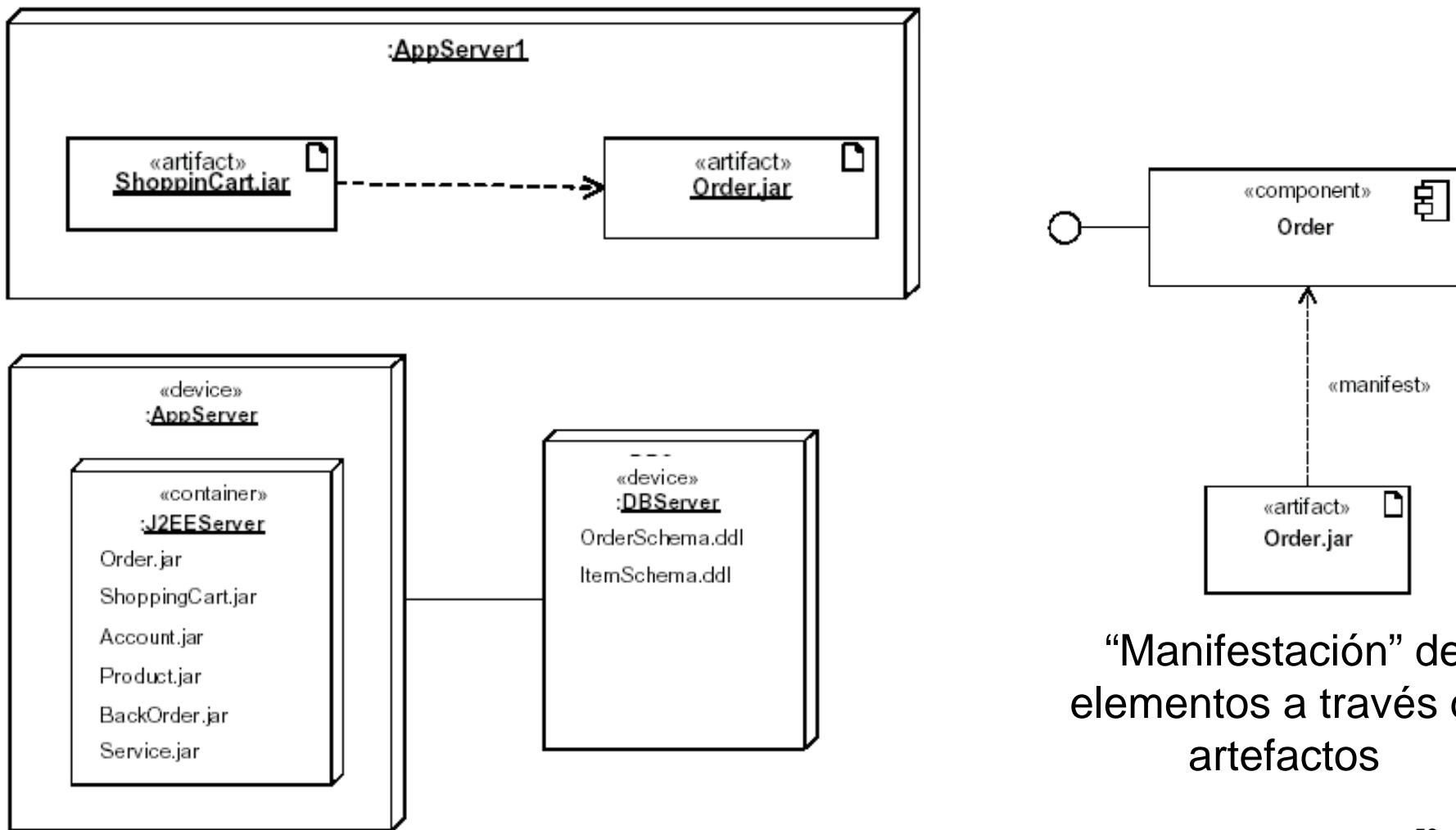
- Definen la arquitectura de ejecución de un sistema.
- Representa la asignación de artefactos software a nodos.



- Nodos = elementos hardware, o entornos de ejecución software.
- Artefactos = elementos concretos (p.e.: ficheros) que son el resultado del proceso de desarrollo.

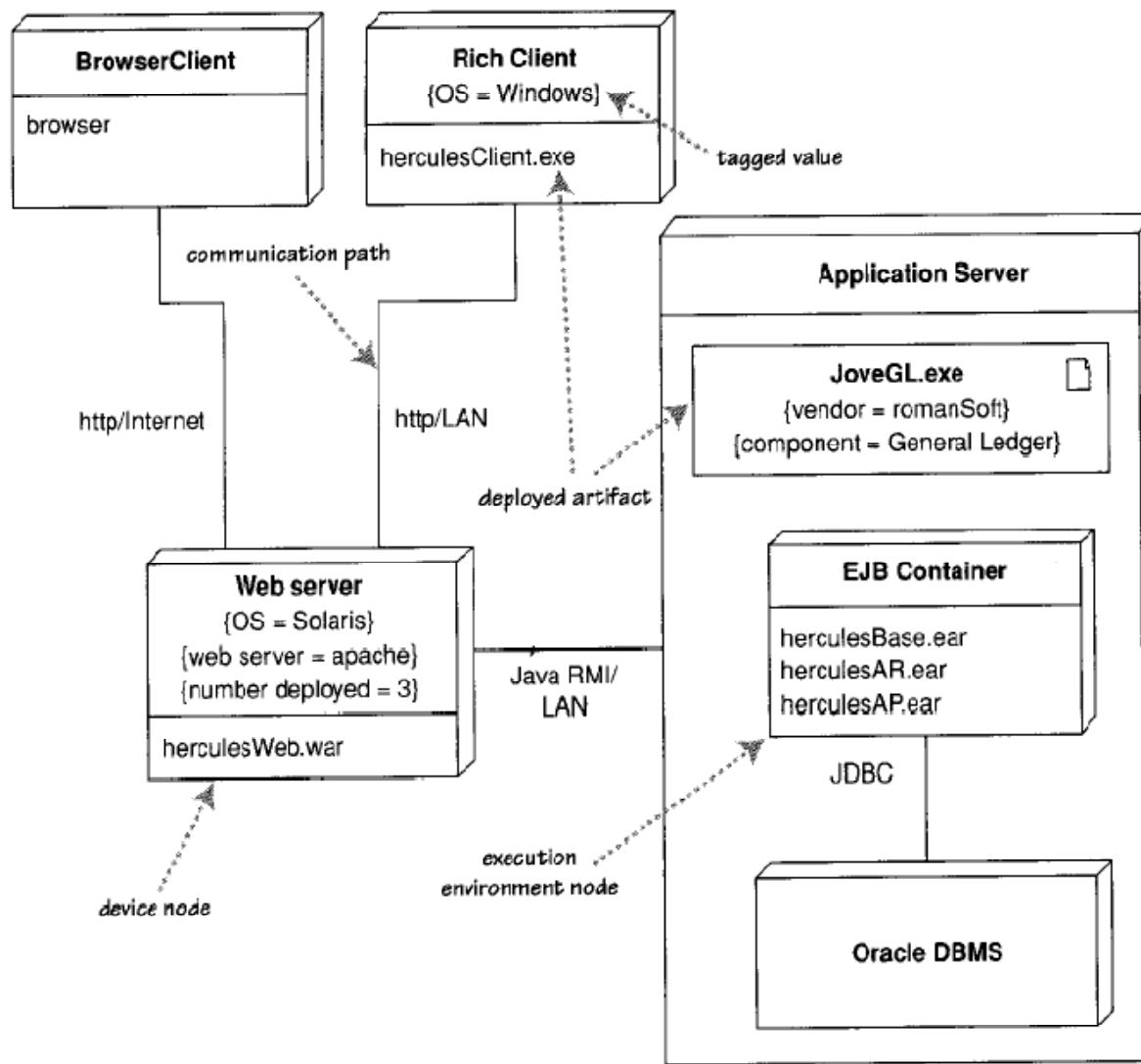
Despliegue

Relaciones entre artefactos.



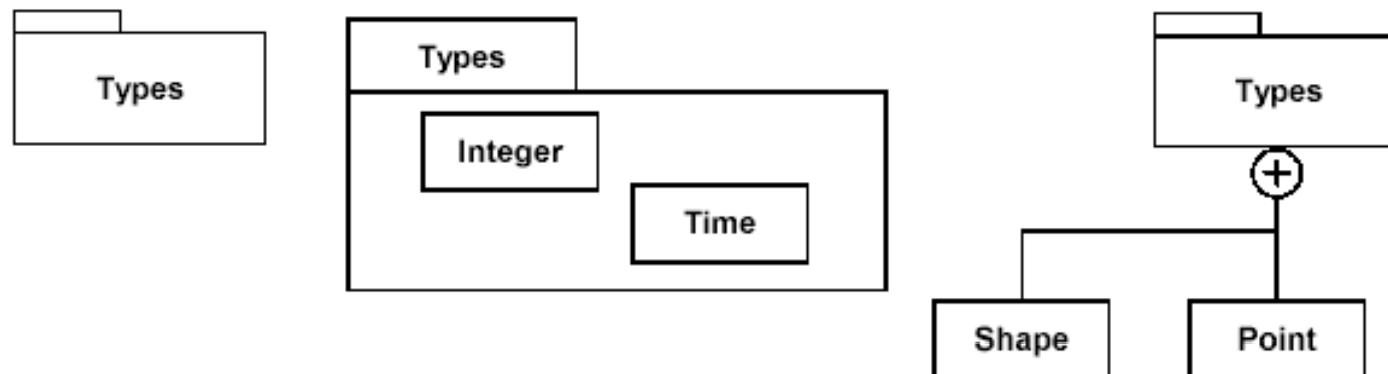
Despliegue

Ejemplo.



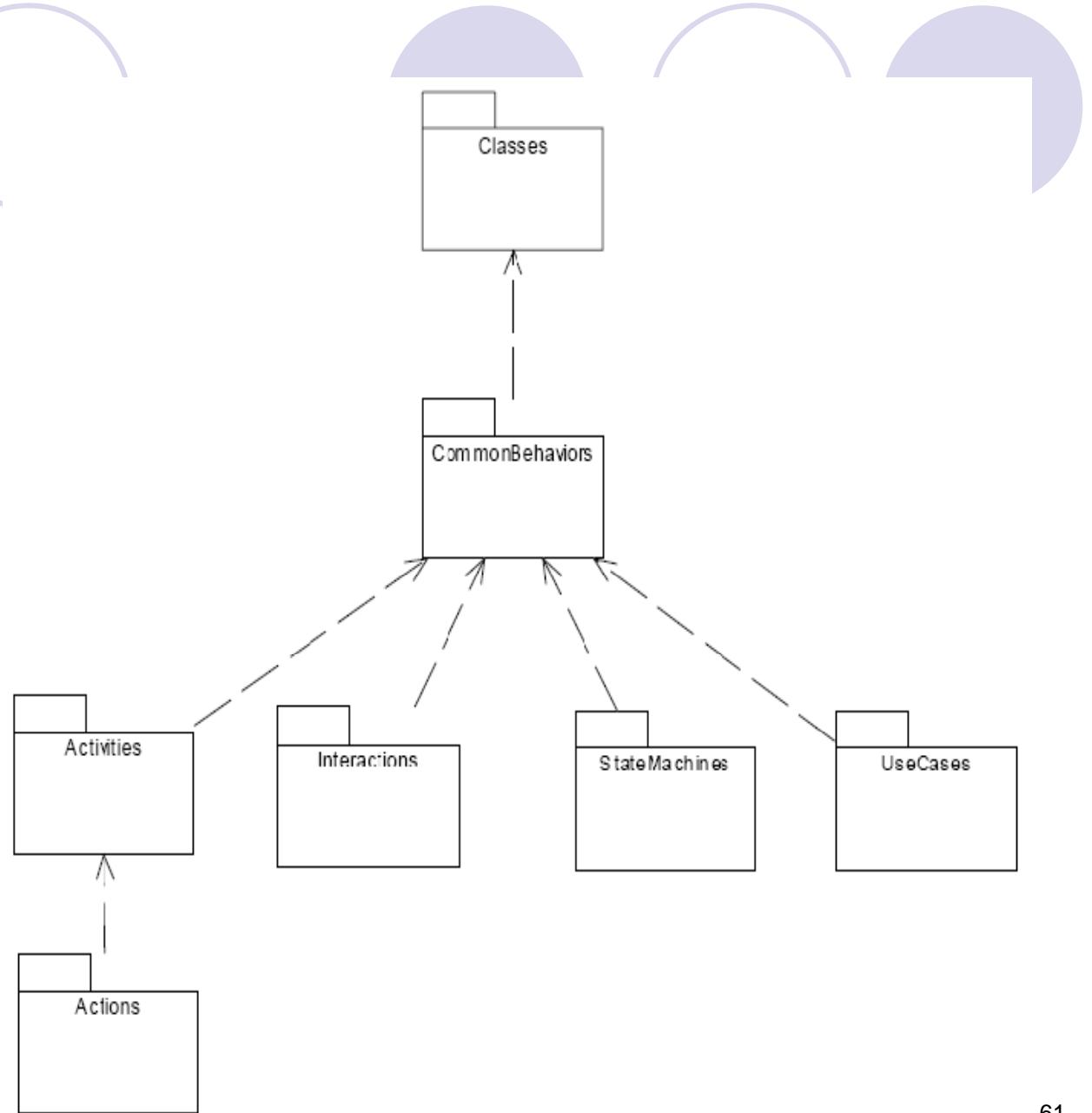
Paquetes

- Un paquete es un contenedor que agrupa elementos relacionados.
- Los diagramas de paquetes muestran la estructura de alto nivel de la aplicación.



Paquetes

Ejemplo.



Indice

- Diagramas de Casos de Uso.
- Diagramas de Estructura.
- **Diagramas de Comportamiento.**
 - Diagramas de Interacción.
 - Diagramas de Comunicación.
 - Diagramas de Secuencia.
 - Diagramas de visión de conjunto de la interacción.
 - Diagramas de tiempo.
 - Diagramas de Estados.
 - Diagramas de Actividad.
- OCL
- Herramientas.
- Ejemplos.
- Bibliografía.

Diagramas de Interacción

- El comportamiento se representa a través de **colaboraciones**:
 - Colección de objetos:
 - Instancias
 - Roles
 - Especifica cómo los objetos y las asociaciones cooperan
 - En un contexto dado
 - Con un propósito específico

Diagramas de Interacción

- El patrón de mensajes dentro de una colaboración es una **interacción**:
 - Mensajes: señales, invocaciones, interacciones implícitas a través de condiciones y eventos temporales.
 - Especifica secuencia de mensajes para cumplir el objetivo de la colaboración
 - Para especificar la interacción, es necesario especificar primero la colaboración
 - Semántica basada en *trazas*.

Diagramas de Interacción

- **Diagrama de Comunicación**
 - Muestra un contexto y una interacción.
- **Diagrama de Secuencia**
 - Representación explícita de la secuencia de comunicaciones, eje temporal.
 - Es más apropiado para aplicaciones de Tiempo Real y escenarios complicados

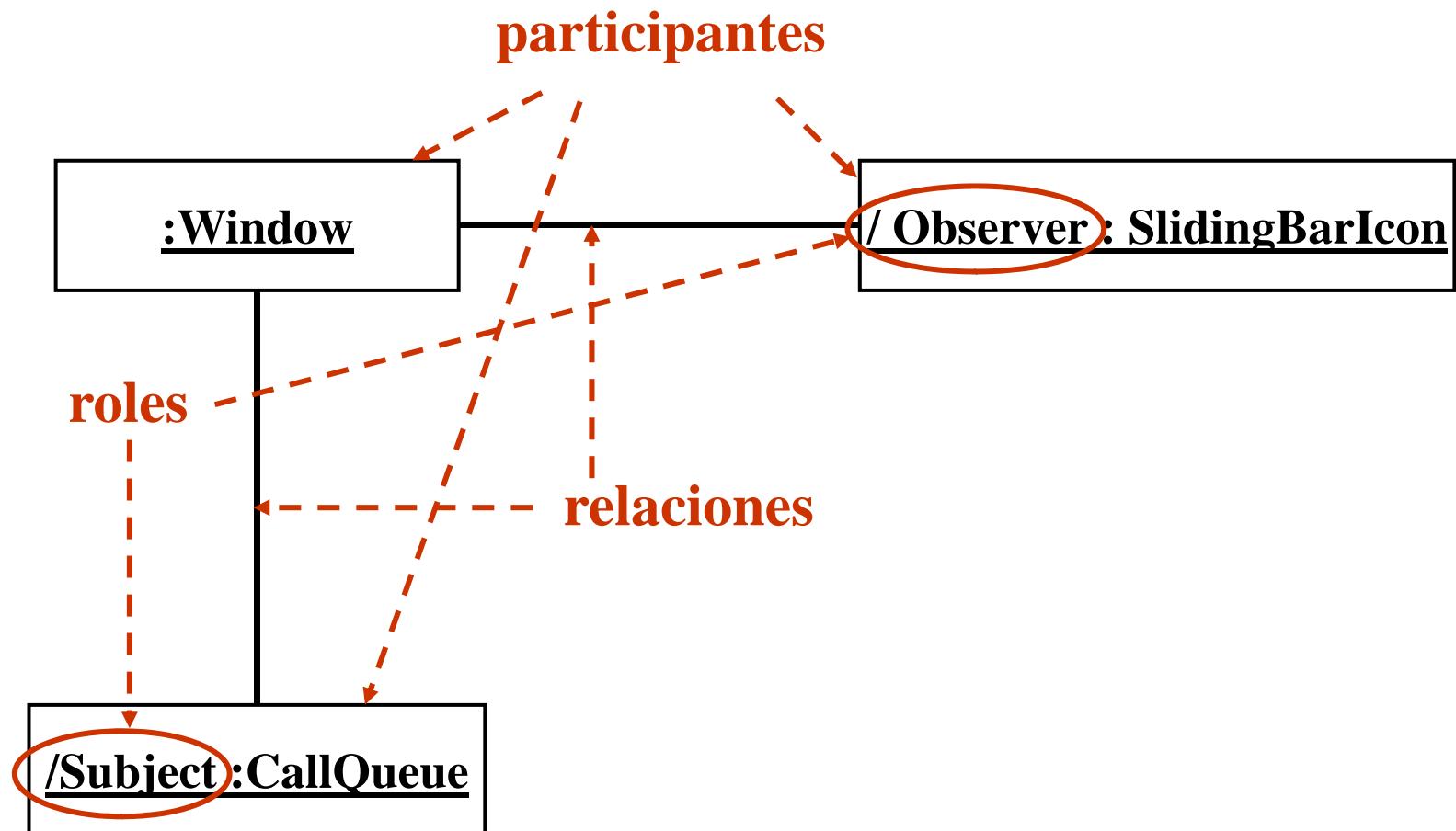
Diagramas de Interacción

- Estructura de los participantes
 - Diagramas de Comunicación
- Patrones de comunicación
 - Diagramas de Comunicación
 - Diagramas de Secuencia
- Temporización de la comunicación.
 - Diagramas de Secuencia
 - Diagramas de Tiempo.
- Estructuración de las interacciones
 - Diagrama de visión de conjunto de la interacción.

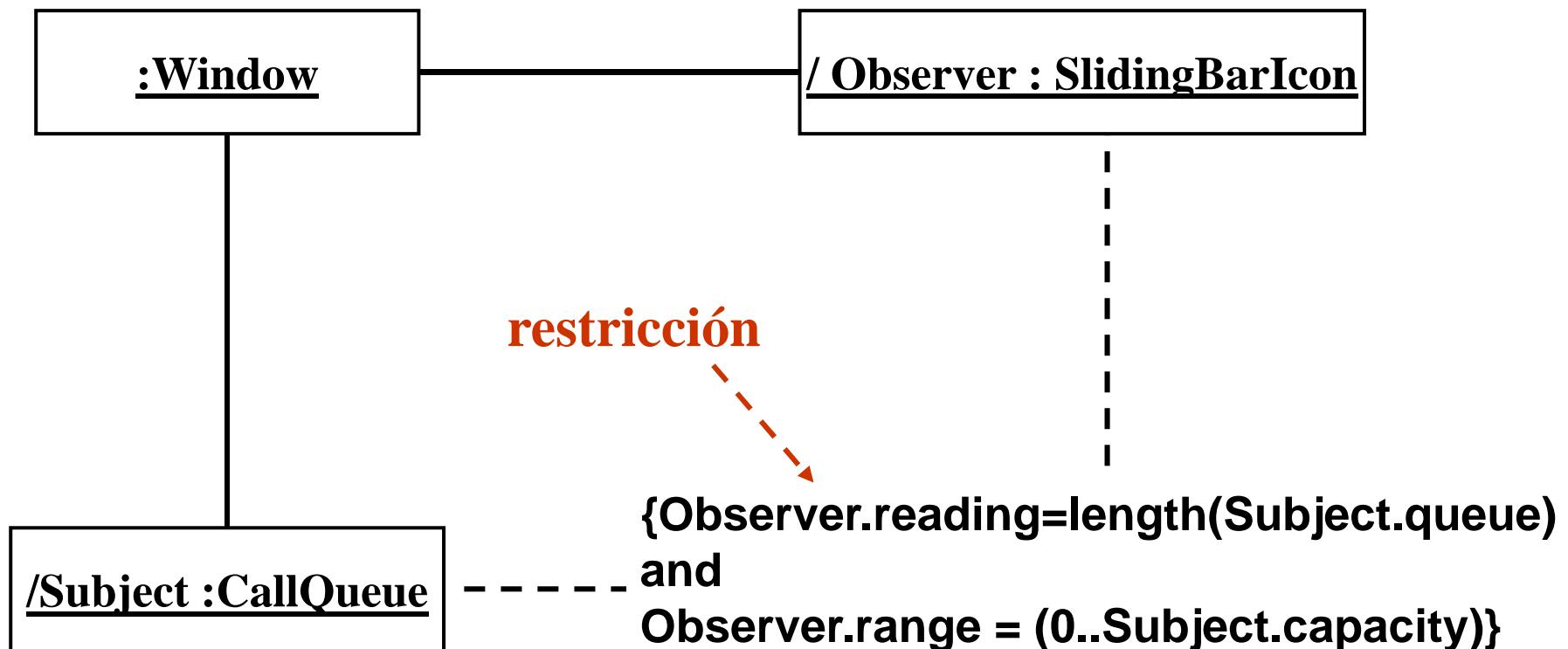
Diagramas de Comunicación

- Representa los objetos (roles o instancias) necesarios para una interacción y sus relaciones.
- Puede también representar la secuencia de mensajes
 - Especifica el orden relativo mediante números
- Similar a un diagrama de objetos, muestra el contexto necesario para una colaboración.

Diagramas de Comunicación

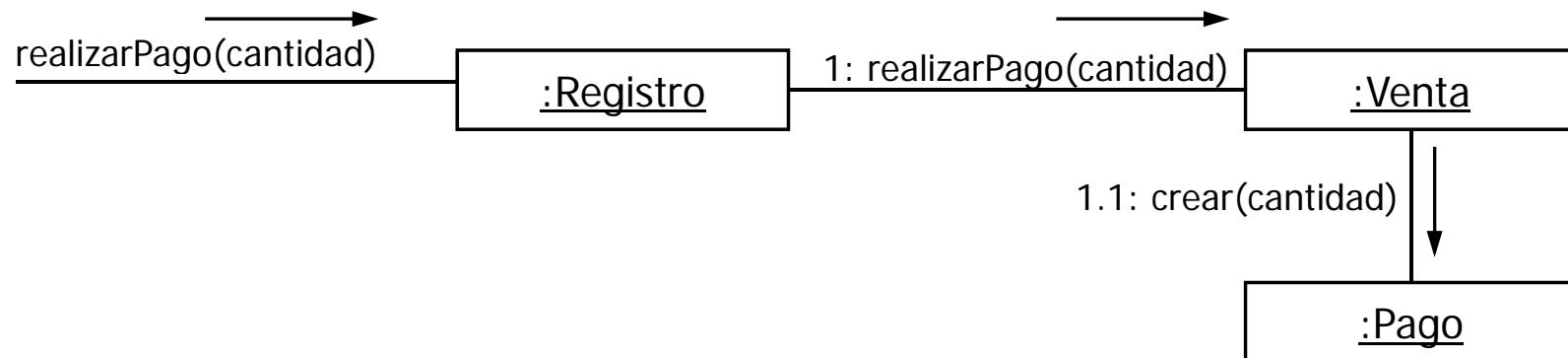


Diagramas de Comunicación

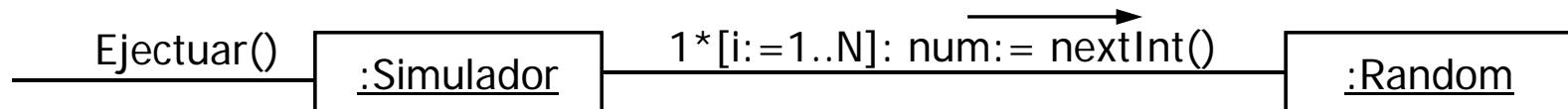


Diagramas de Comunicación

Llamadas anidadas:



Iteraciones:



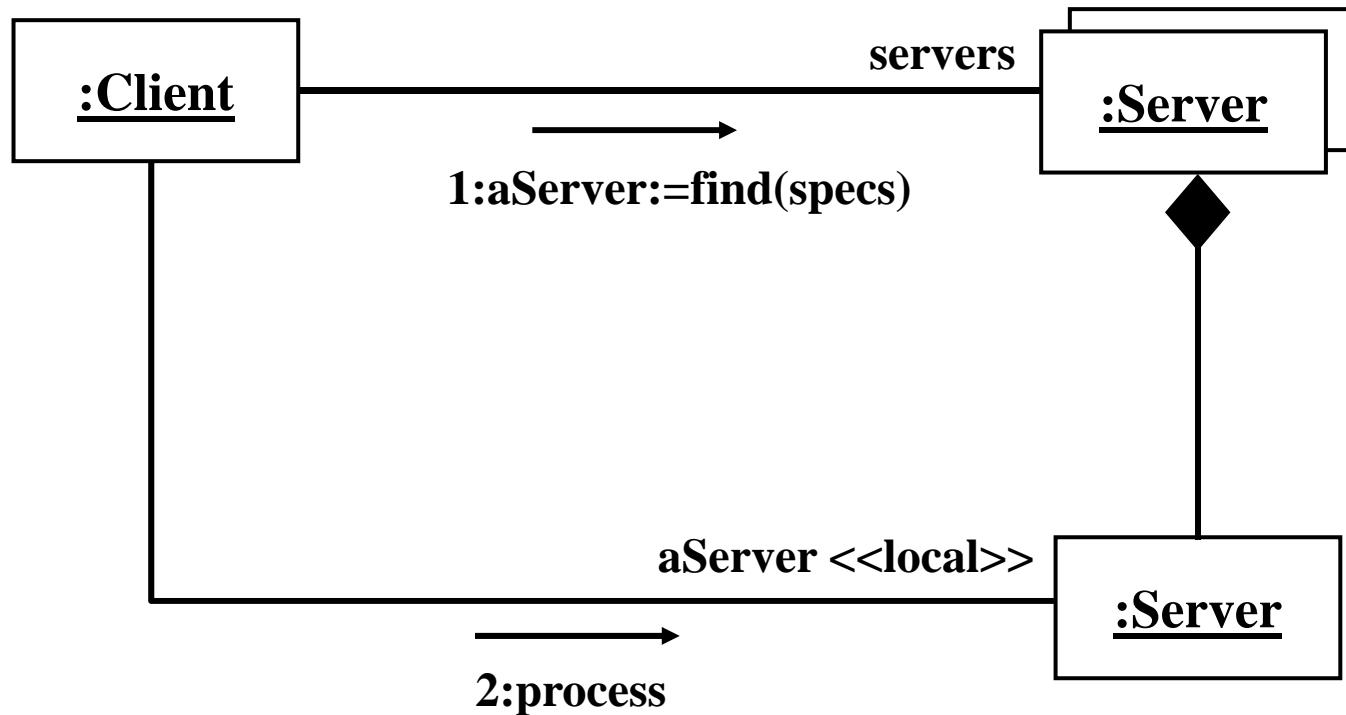
Diagramas de Comunicación

Multiobjetos.

- Representan conjuntos de instancias en el extremo “muchos” de una relación 1:N o M:N
- Una operación sobre cada instancia requiere dos mensajes:
 - Iteración para obtener referencias a las instancias individuales
 - Mensaje a cada instancia, usando la referencia temporal

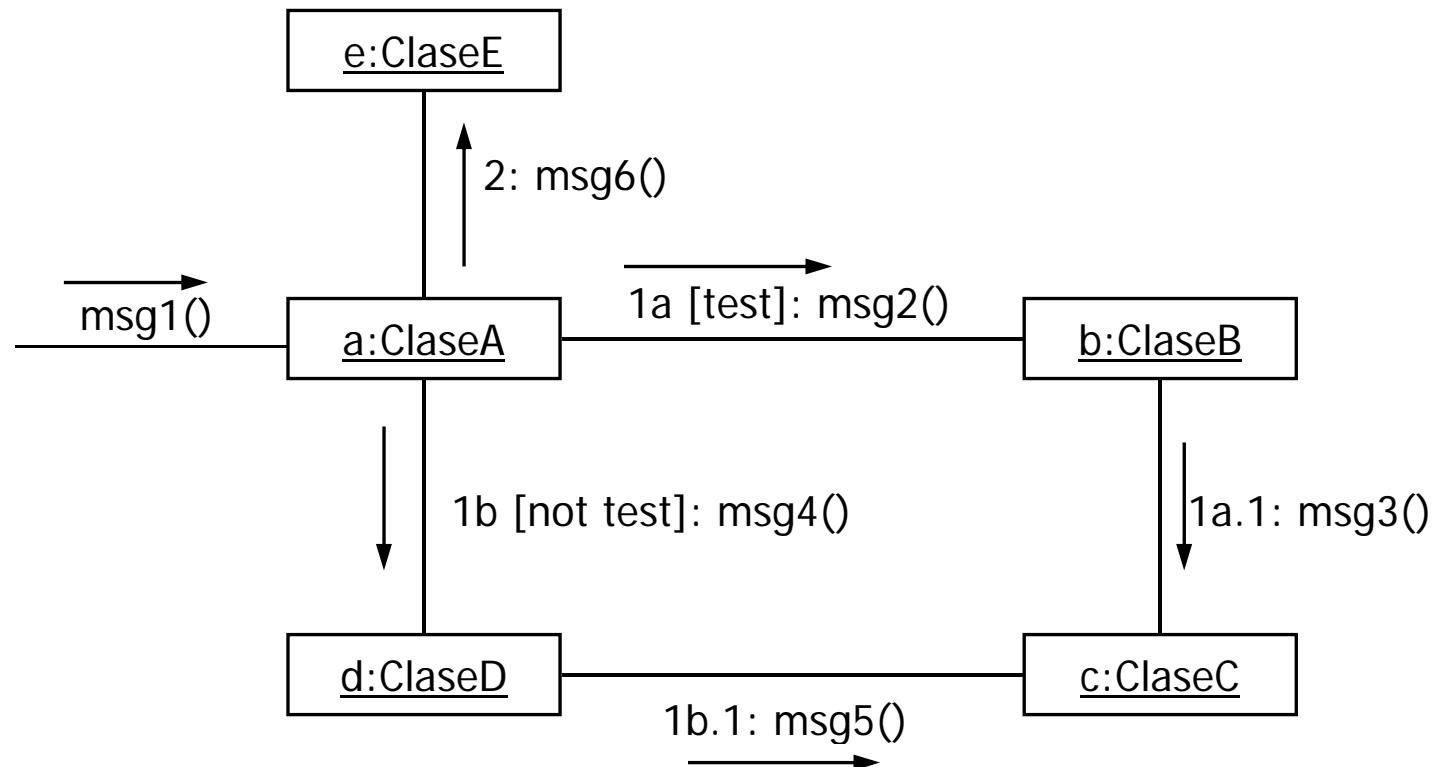
Diagramas de Comunicación

Multiobjetos.



Diagramas de Comunicación

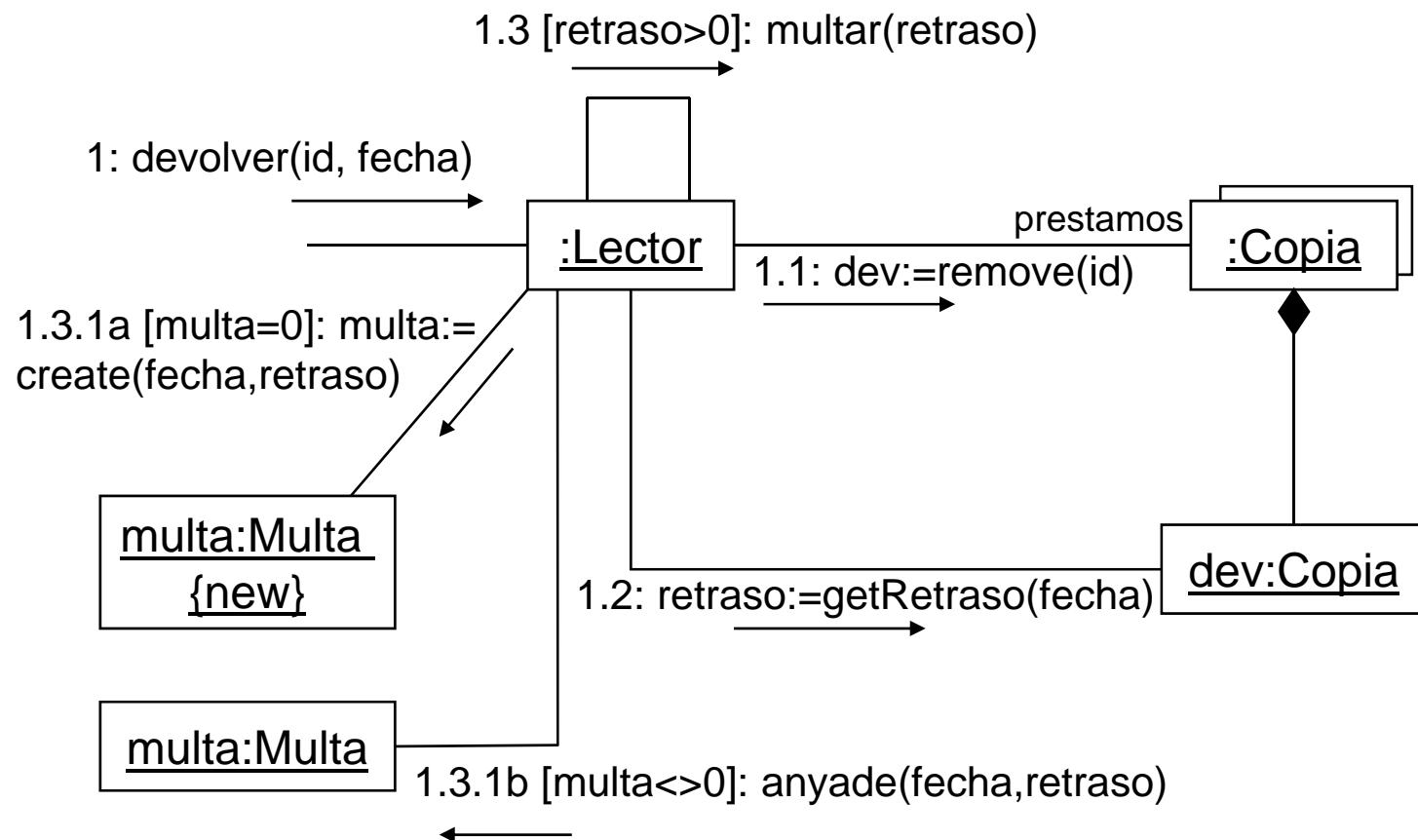
Condiciones.



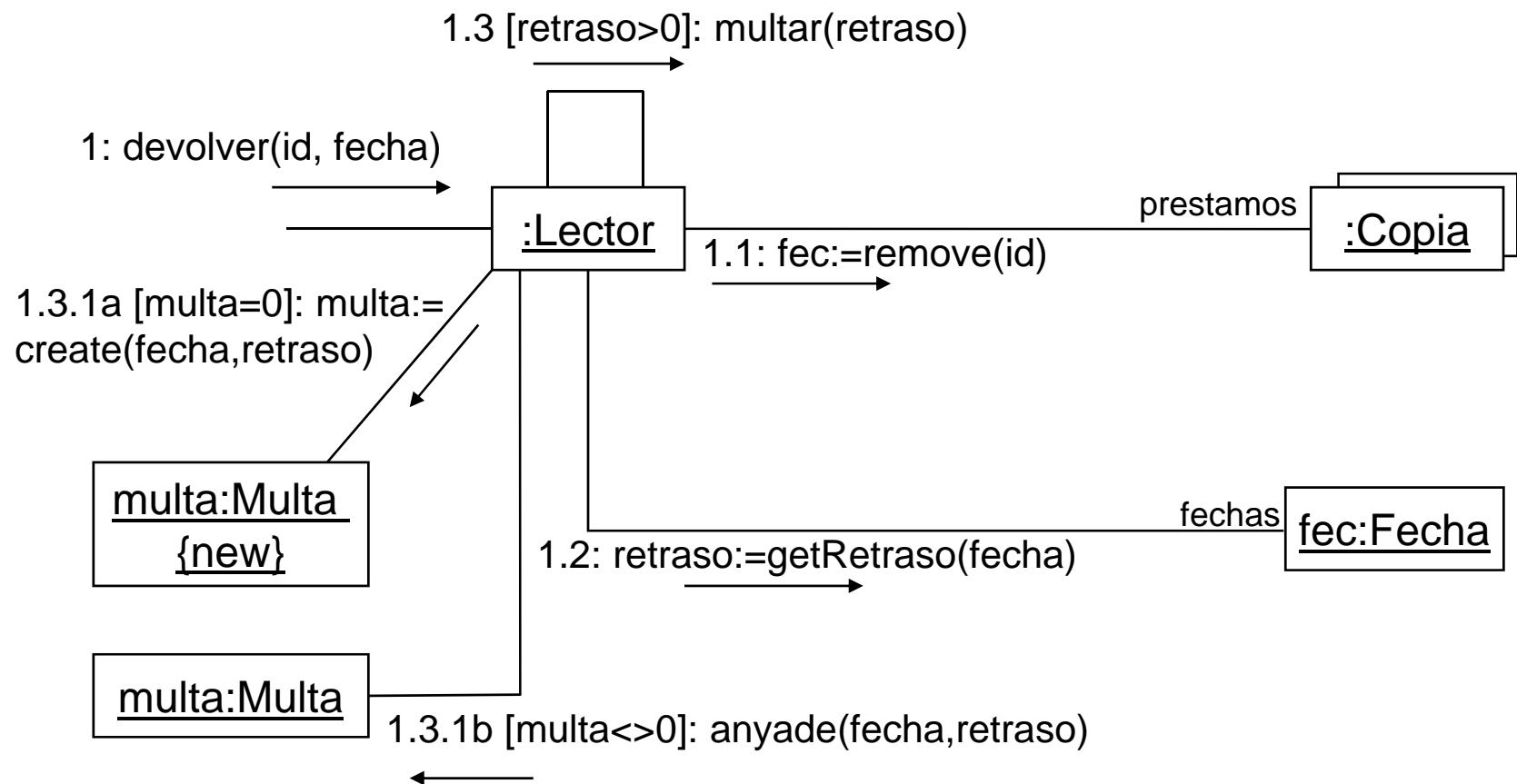
Ejercicio: Biblioteca

- Una biblioteca tiene copias de libros. Estos últimos se caracterizan por su nombre, tipo (novela, teatro, poesía, ensayo), editorial, año y autor.
- Los autores se caracterizan por su nombre, nacionalidad y fecha de nacimiento.
- Cada copia tiene un identificador, y puede estar en la biblioteca, prestada, reservada, con retraso o en reparación.
- Los lectores pueden tener un máximo de 3 libros en préstamo.
- Cada libro se presta un máximo de 30 días, por cada día de retraso, se impone una “multa” de dos días sin posibilidad de coger un libro.
- Realiza el diagrama de colaboración para el método devolver()

Solución



Solución



Diagramas de Comunicación

Etiquetas de las flechas.

predecesor orden-secuencial valor-retorno

:= nombre-mensaje lista-argumentos

- Predecesor

- Lista separada por “,” terminada en “/”
- El mensaje no está habilitado hasta que no ocurran los mensajes en la lista

- Orden en la secuencia

- Lista separada por “.”, terminada en “:”
- Cada término representa nivel de anidamiento procedural
- Iteración: *[cláusula iteración]
- Bifurcación: [condición]

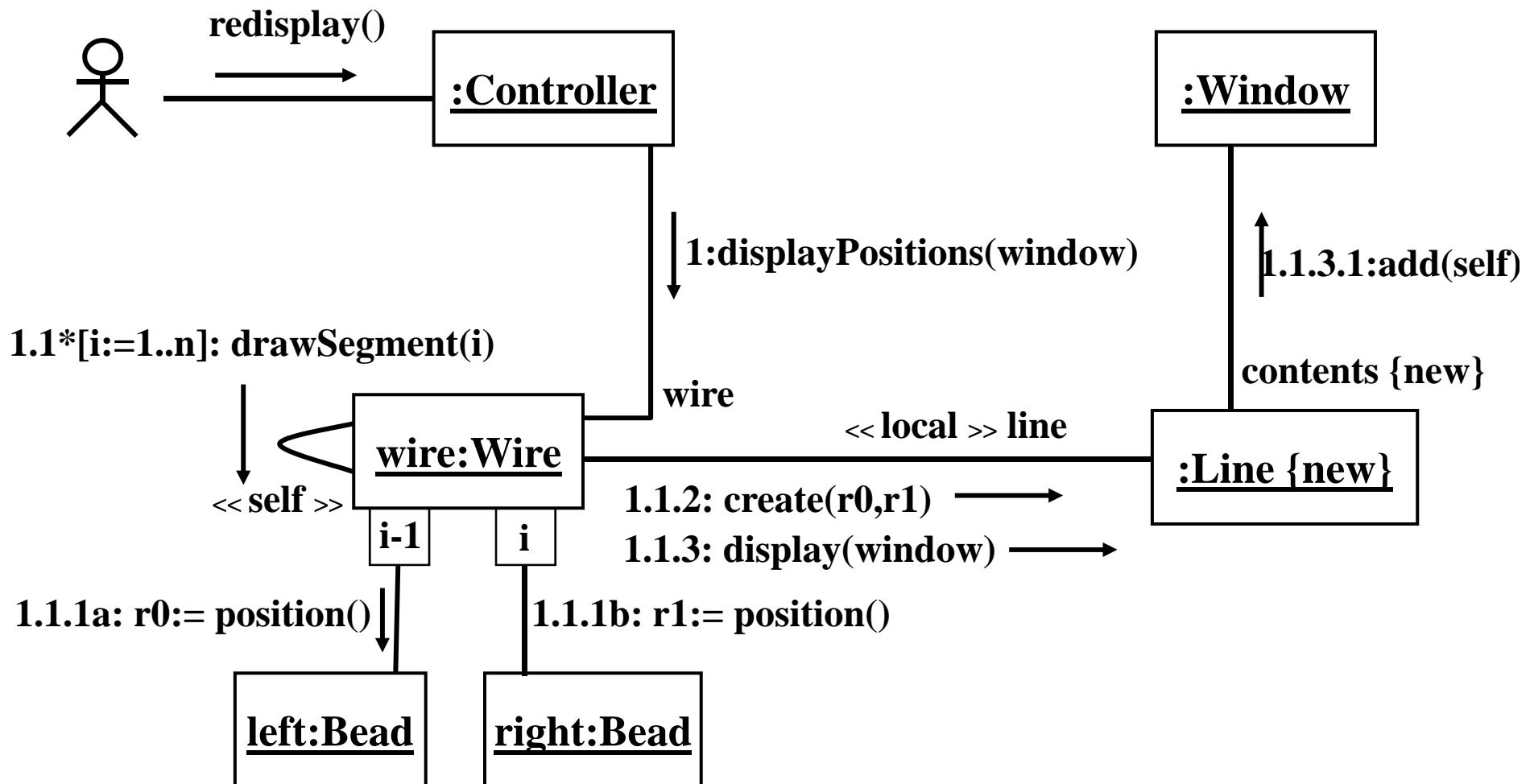
Diagramas de Comunicación

Etiquetas de las flechas.

- Ejemplos:
 - 2: display(x,y)
 - 1.3.1: p:=find(specs)
 - 4 [x<0]: invert(x, color)
 - A3,B4/ C3.1:update()
 - 1..1 *[i:=1..n]: lecturer()

Diagramas de Comunicación

Ejemplo.



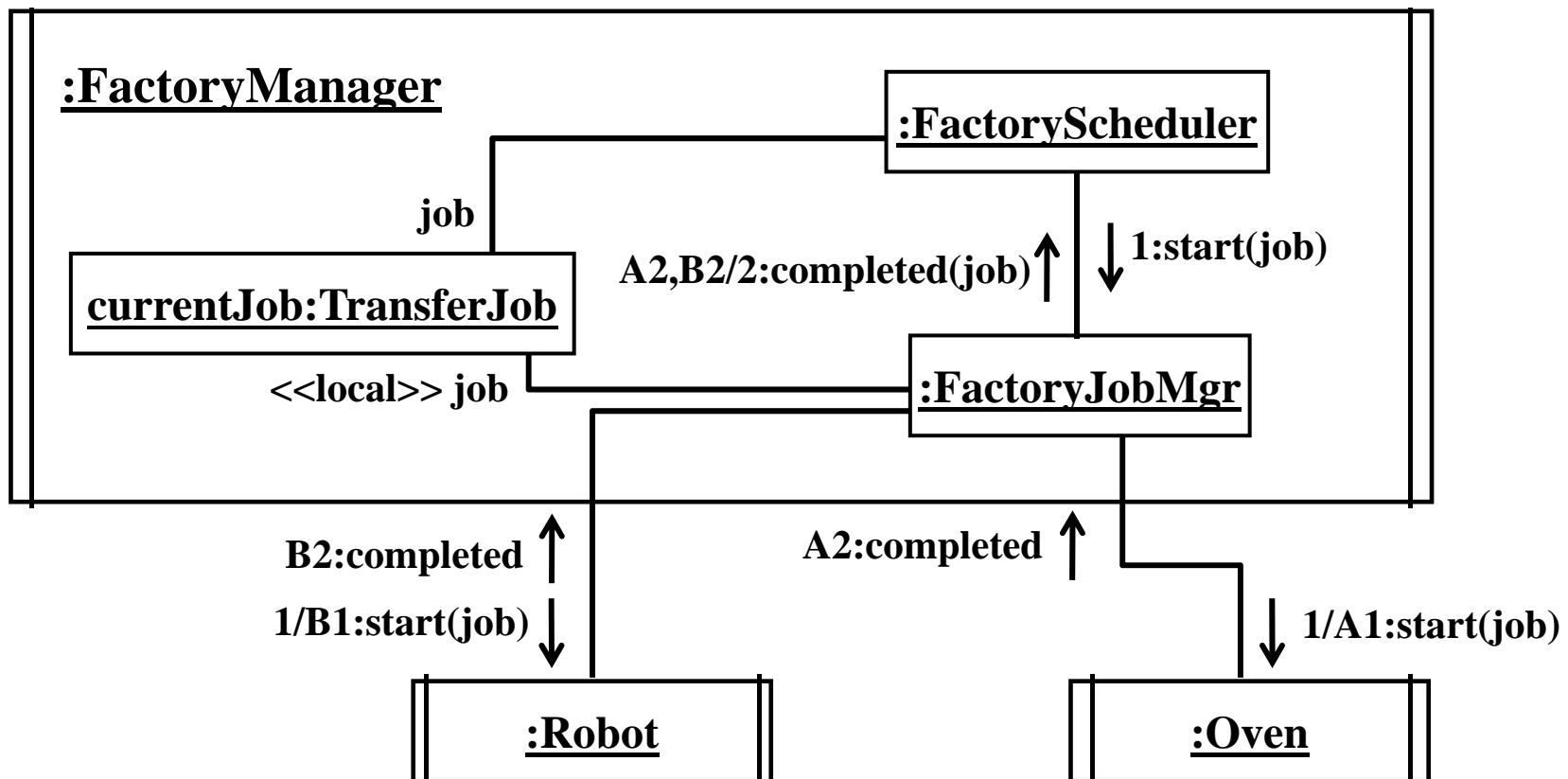
Diagramas de Comunicación

Objetos Activos.

- Poseen su propio hilo de control
- Un objeto pasivo sólo almacena datos
 - Puede enviar mensajes mientras procesa un pedido (mensaje) que haya recibido
- Los objetos activos se representan frecuentemente con componentes internas

Diagramas de Comunicación

Objetos Activos. Ejemplo.



Diagramas de Comunicación

Objetos Activos. Ejemplo.

Tipos de Flujos de Control

→	Llamada a procedimiento u otra forma de llamada con anidamiento de control. La secuencia anidada termina antes de que siga la operación que invocó. Puede usarse para procesos concurrentes cuando el mensaje es síncrono.
→	Comunicación asincrónica, sin anidamiento de control. El objeto que envía no se detiene a esperar respuesta.
→	Retorno de una llamada a procedimiento. Esta flecha puede omitirse si queda claro por el fin de la activación.

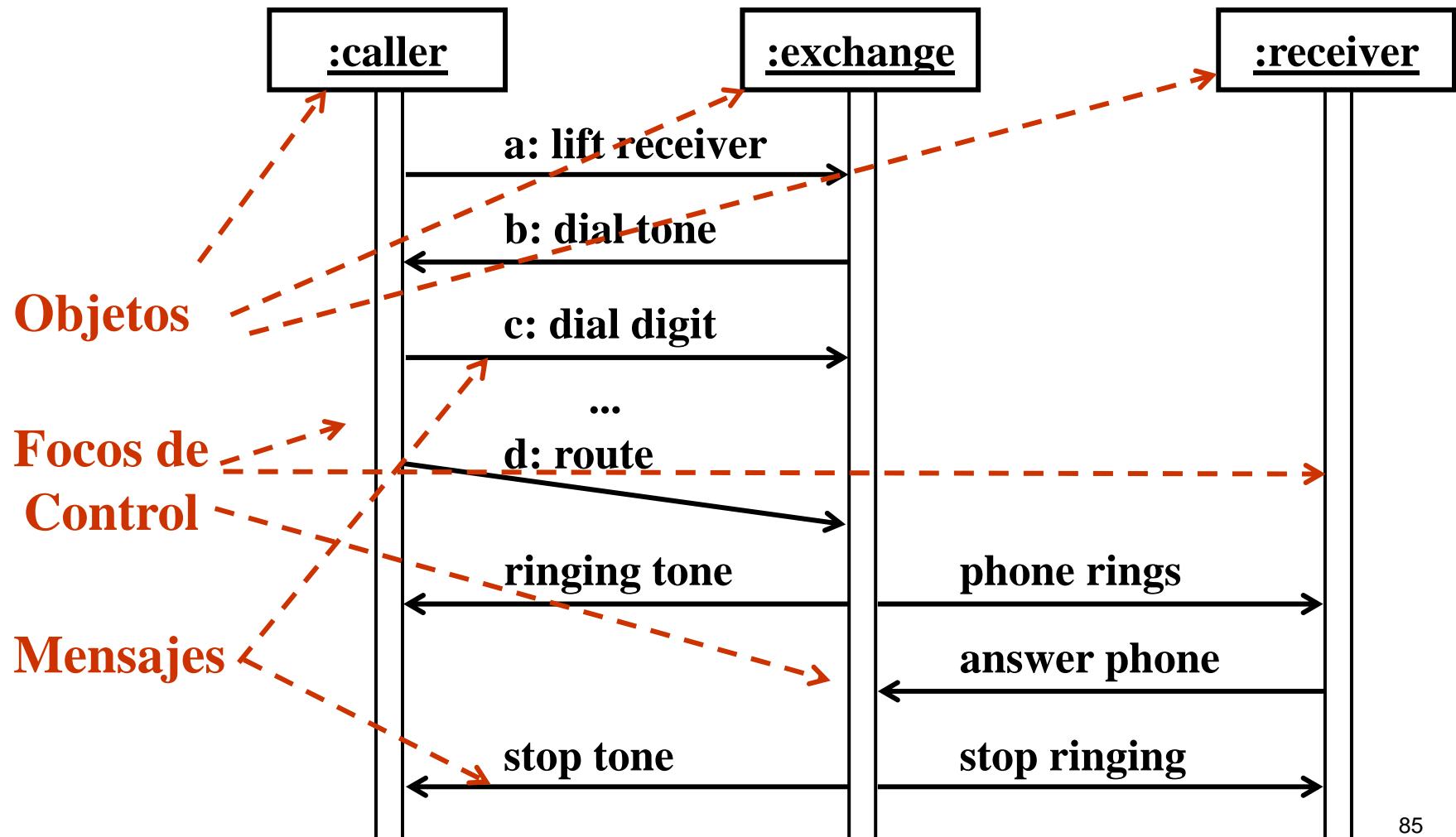
Diagramas de Secuencia

- Representa conjunto de mensajes entre roles (o instancias) en una interacción
- Dos dimensiones:
 - Tiempo (generalmente vertical); puede ser una escala si el sistema es de tiempo real
 - Diferentes instancias (generalmente horizontal); el orden relativo no tiene importancia
- Se muestra la existencia y duración de las instancias, pero no sus relaciones

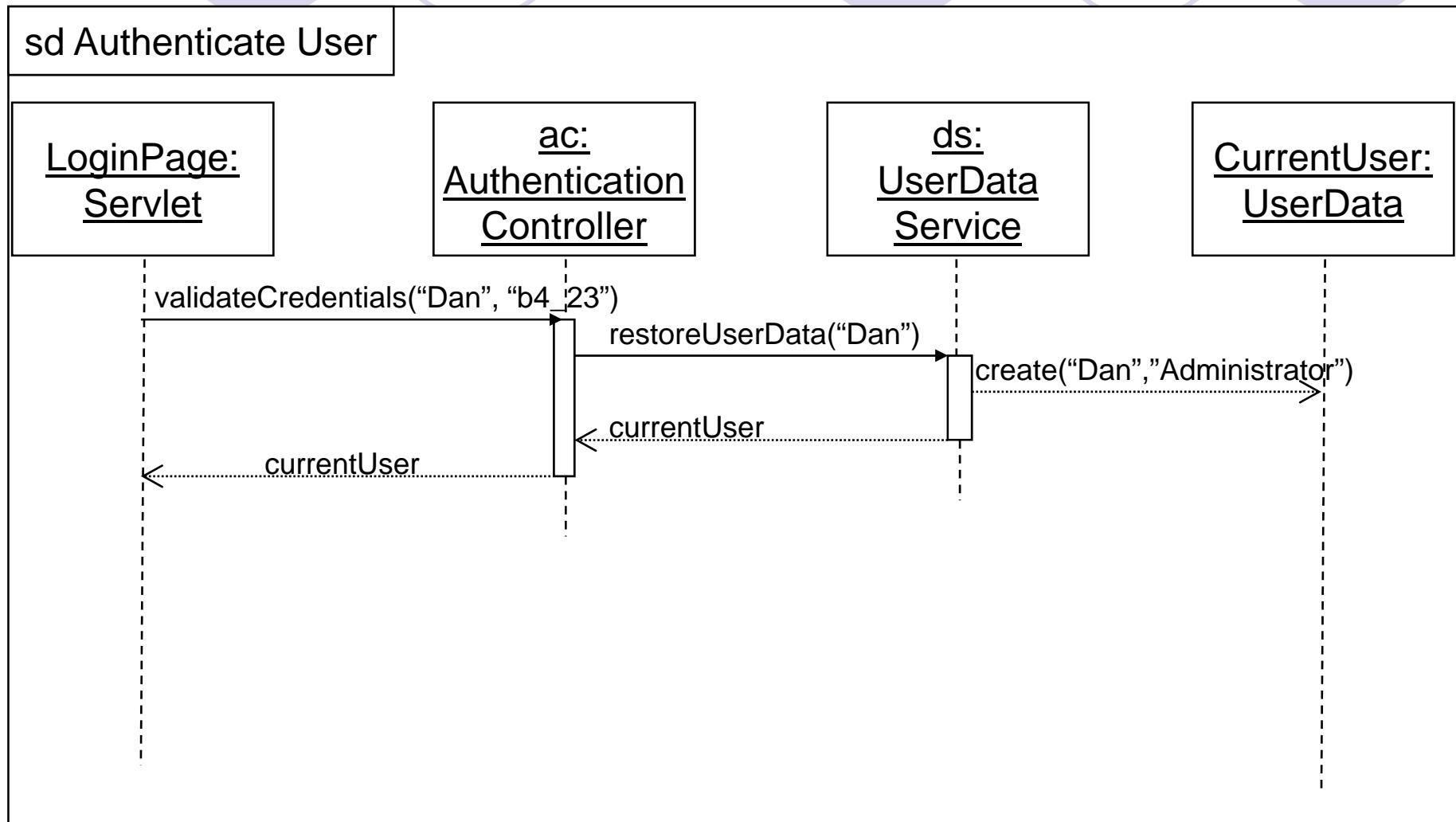
Diagramas de Secuencia

- Traz: secuencia de ocurrencias de eventos $\langle e_1 \ e_2 \ \dots e_n \rangle$
- La semántica de una interacción es un par de conjuntos de trazas (válidas e inválidas) $[P, I]$.
- Pueden existir trazas no incluidas en los dos conjuntos anteriores.
- Equivalencia de interacciones, si sus conjuntos de trazas son iguales.
- Una interacción se puede especializar: se añaden más trazas al conjunto.

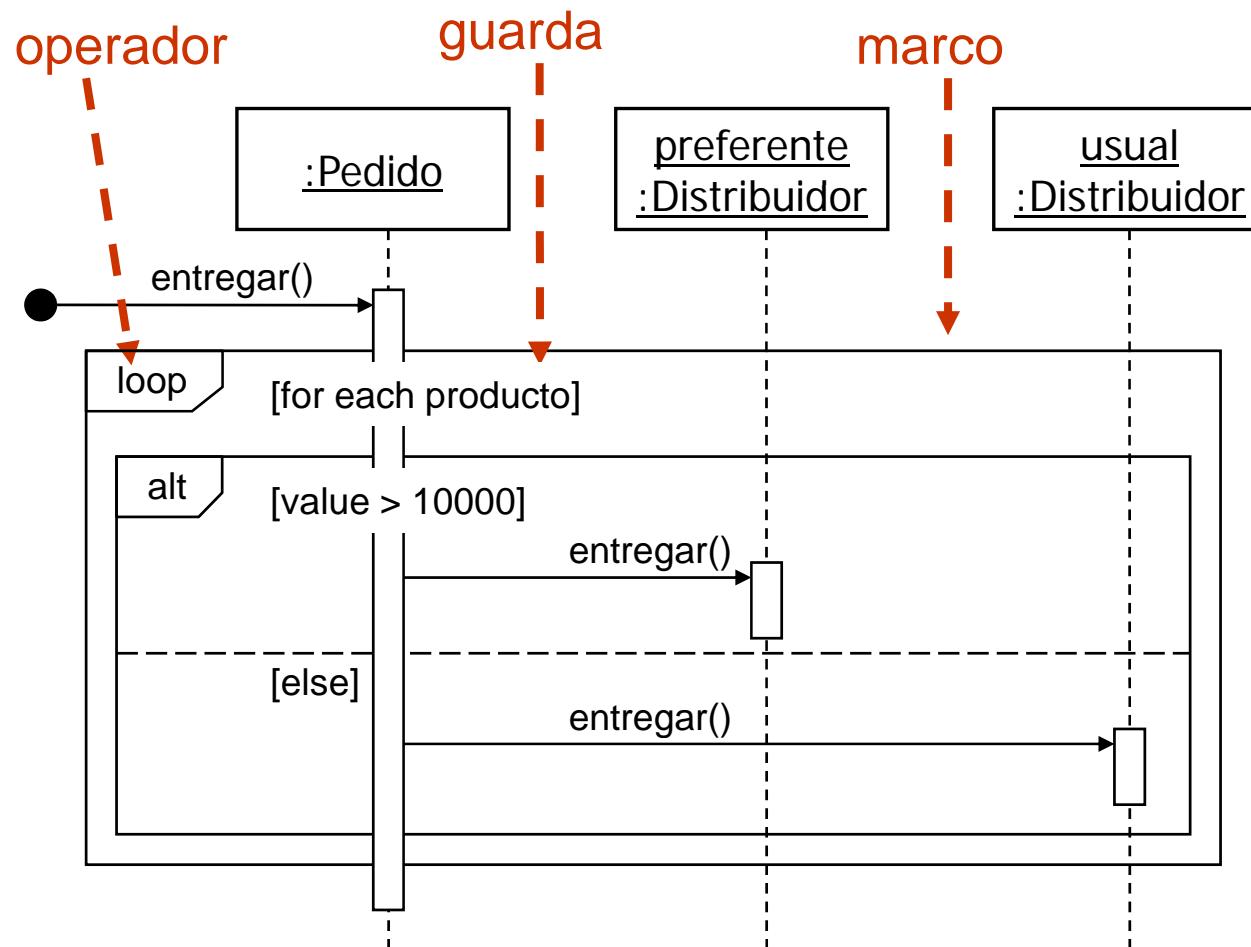
Diagramas de Secuencia



Diagramas de Secuencia

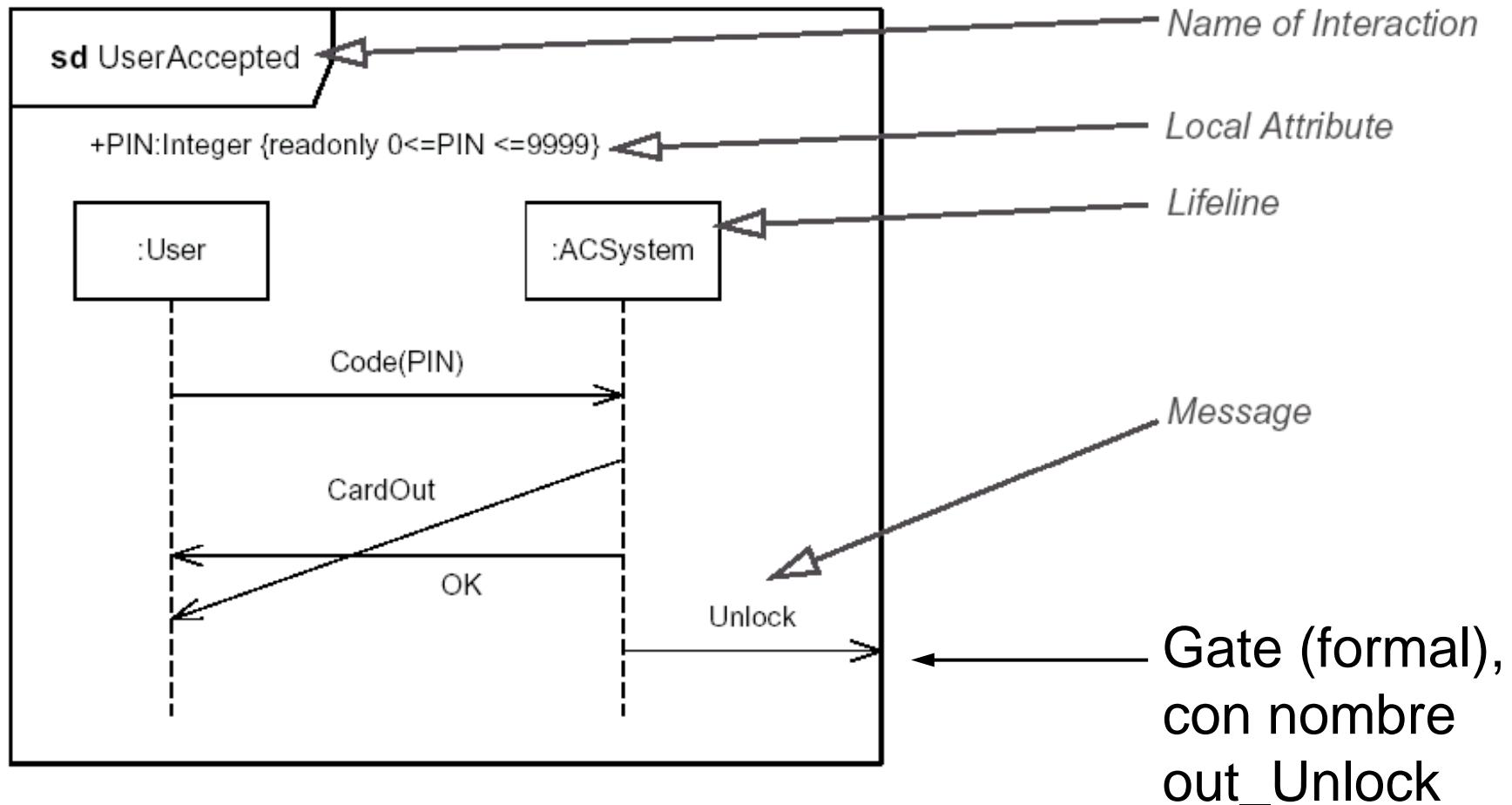


Diagramas de Secuencia



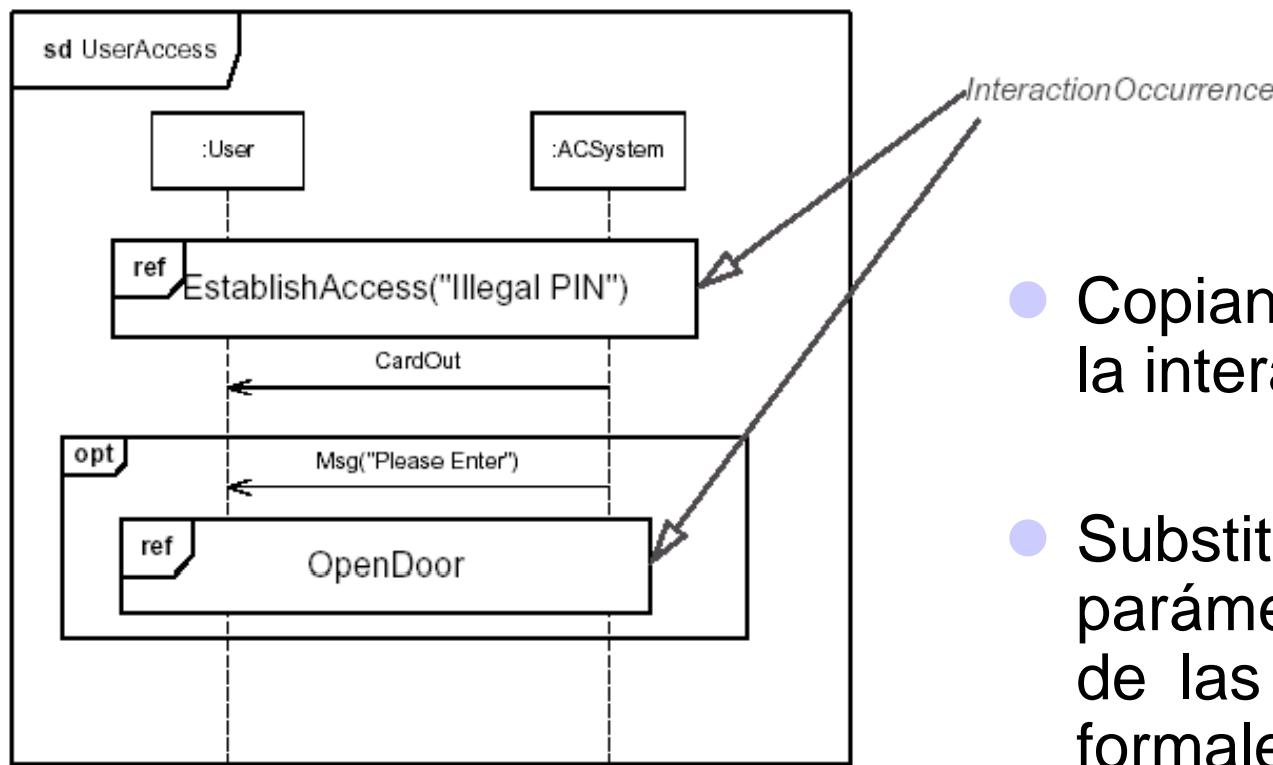
```
procedure entregar()
foreach producto:
    if producto.value>10000
        preferente.entregar()
    else
        usual.entregar()
    end if
end for
end procedure
```

Diagramas de Secuencia



Diagramas de Secuencia

Referencias (Ocurrencias de Interacciones)



- Copian el contenido de la interacción referida.
- Substitución de parámetros y conexión de las puertas (gates) formales y actuales.

Diagramas de Secuencia

Operadores sobre interacciones.

- Fragmentos combinados, operadores (i):
 - Alternativa (*alt*).
 - Elección (mediante una guarda) de una interacción.
 - Aserción (*assert*).
 - La secuencia especificada por el operador es la única válida.
 - Opción (*opt*).
 - Equivalente a un operador alt con un solo fragmento.
 - Ruptura (*break*).
 - El operando se ejecuta en lugar del resto de la interacción englobada en el fragmento “padre”.
 - Paralelo (*par*).
 - Mezcla de las trazas de los operandos (cualquier *entrelazado* es válido mientras preserve el orden de los eventos de cada operando).

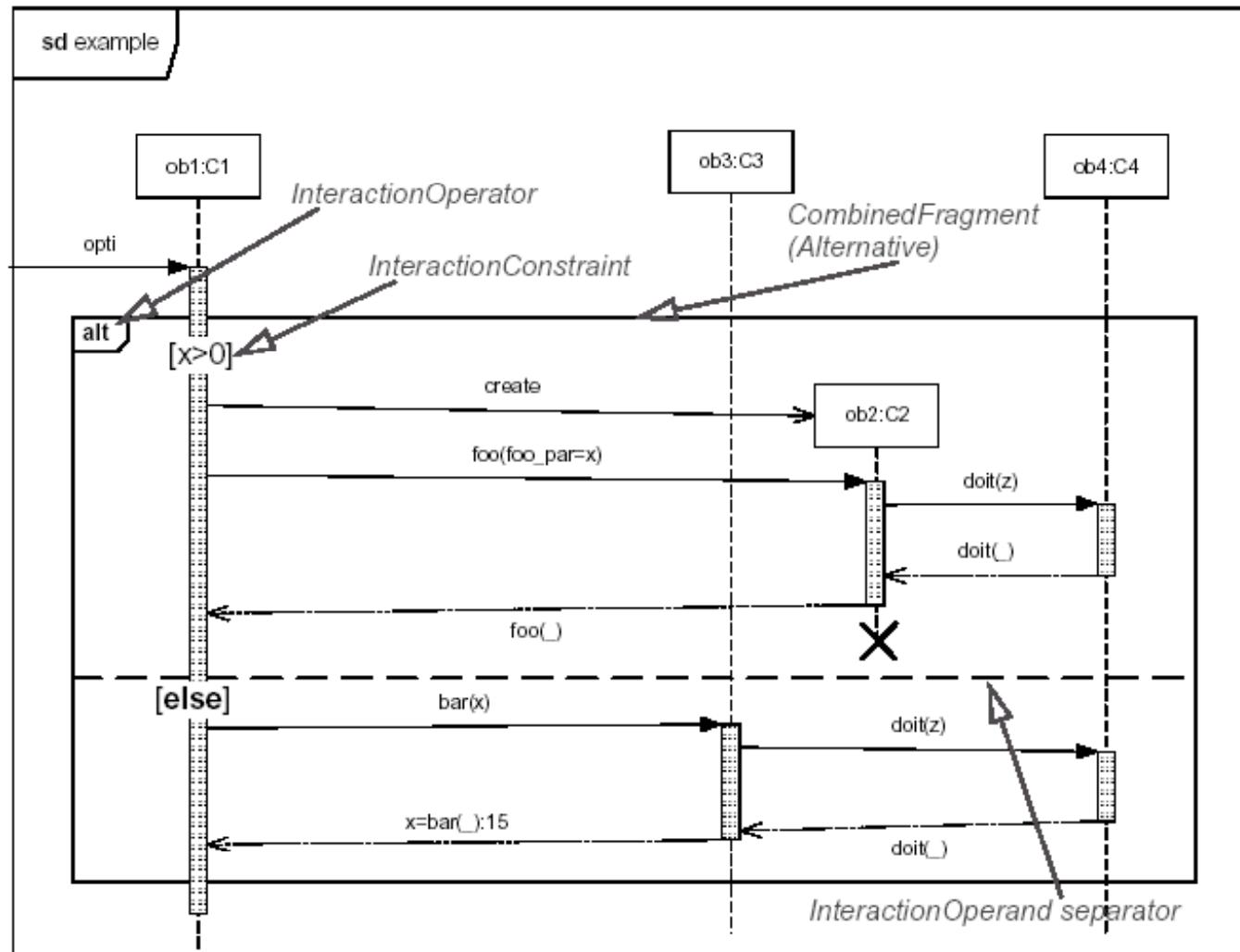
Diagramas de Secuencia

Operadores sobre interacciones.

- Secuenciación débil (*seq*).
 - Define un conjunto de trazas que cumple:
 - 1. Se mantiene el orden de eventos de los operandos
 - 2. Eventos de otras líneas de vida de otros operandos pueden venir en cualquier orden.
 - 3. Eventos de la misma línea de vida de otros operandos se ordenan de tal manera que cualquier evento del primer operando va antes que el del segundo.
- Secuenciación estricta (*strict*).
 - Secuenciación estricta en el orden de los eventos de los operandos.
- Negativa (*neg*).
 - Define trazas inválidas.
- Región crítica (*critical*).
 - Los eventos del operando no pueden mezclarse con ningún otro.

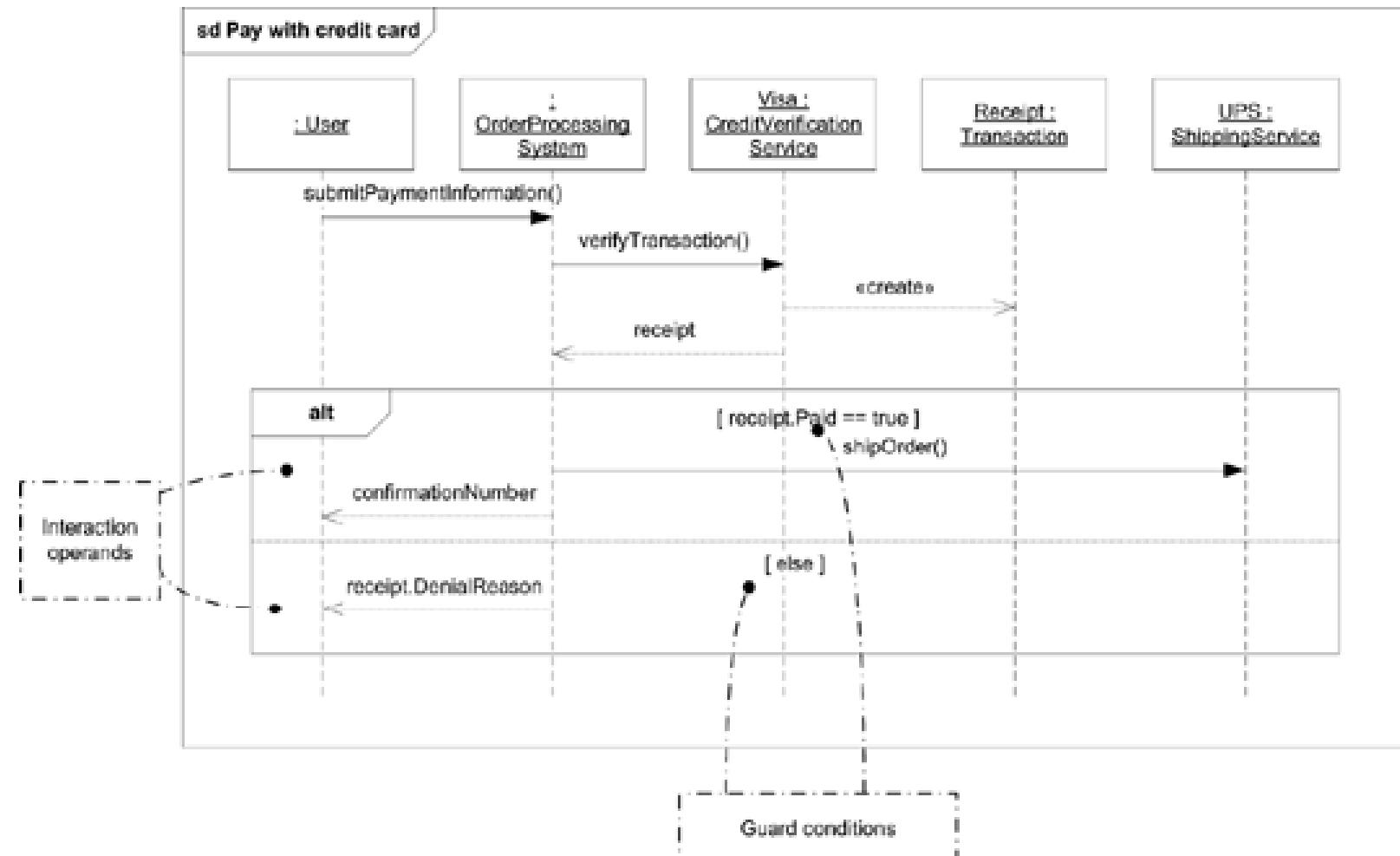
Diagramas de Secuencia

Operadores sobre interacciones. Alternativa.



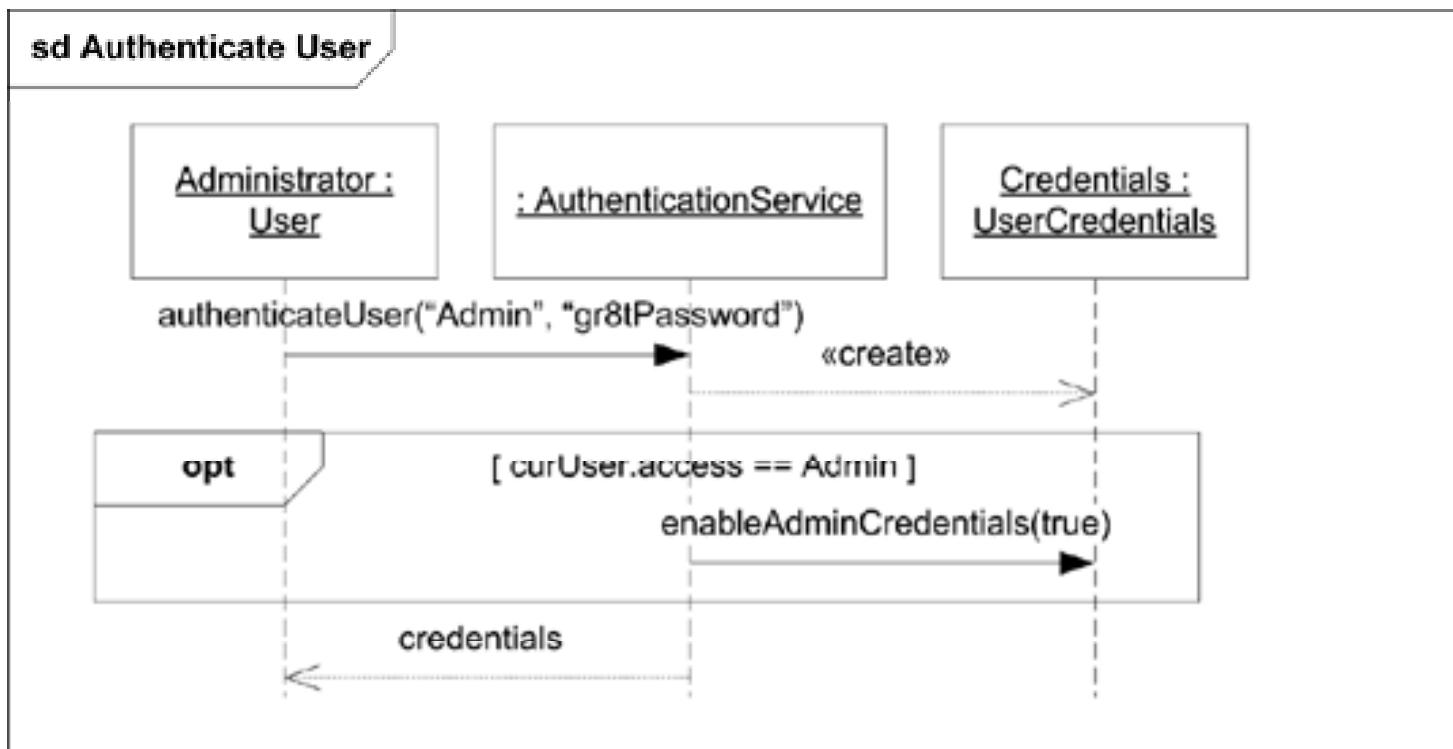
Diagramas de Secuencia

Operadores sobre interacciones. Alternativa.



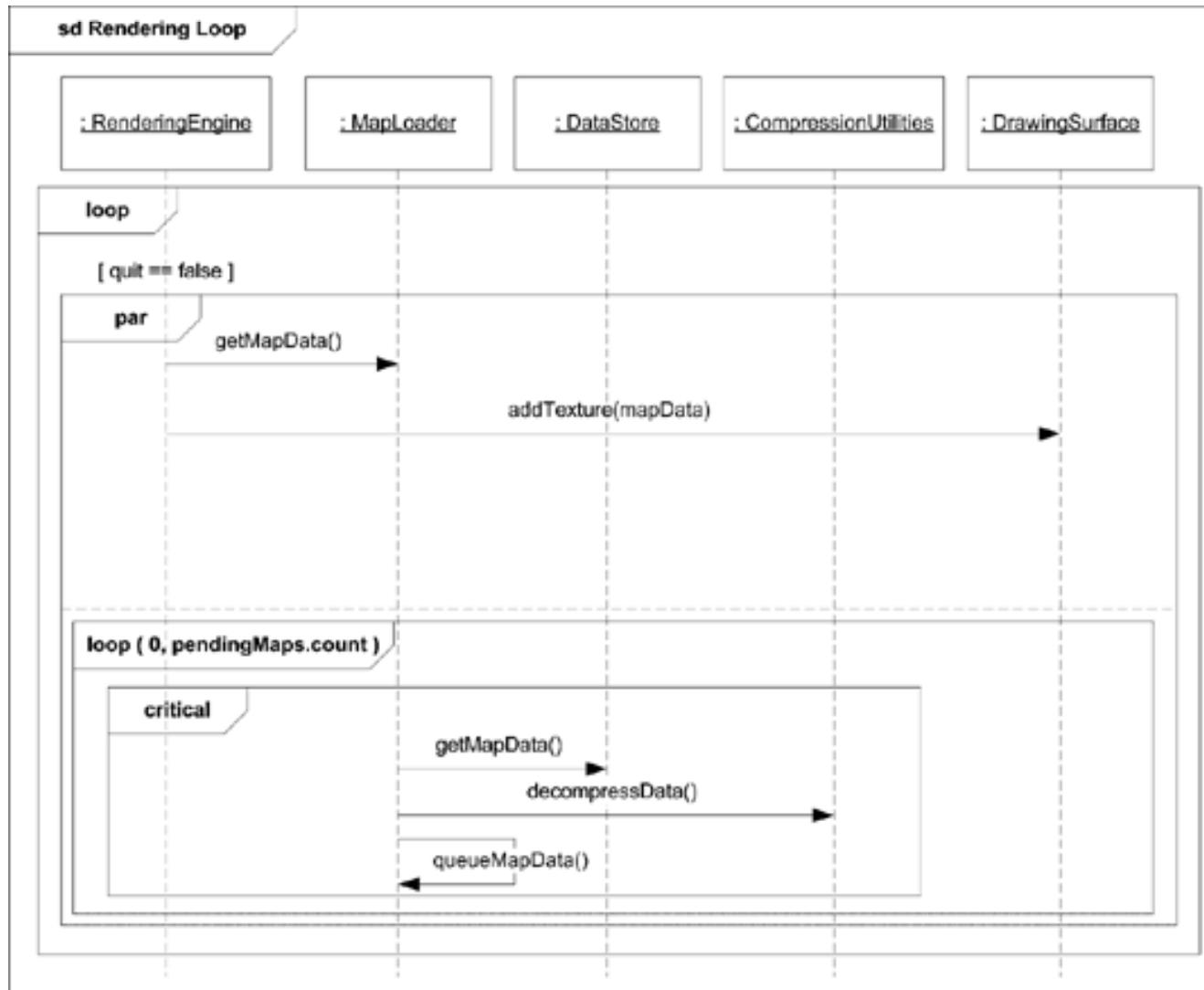
Diagramas de Secuencia

Operadores sobre interacciones. Opción.



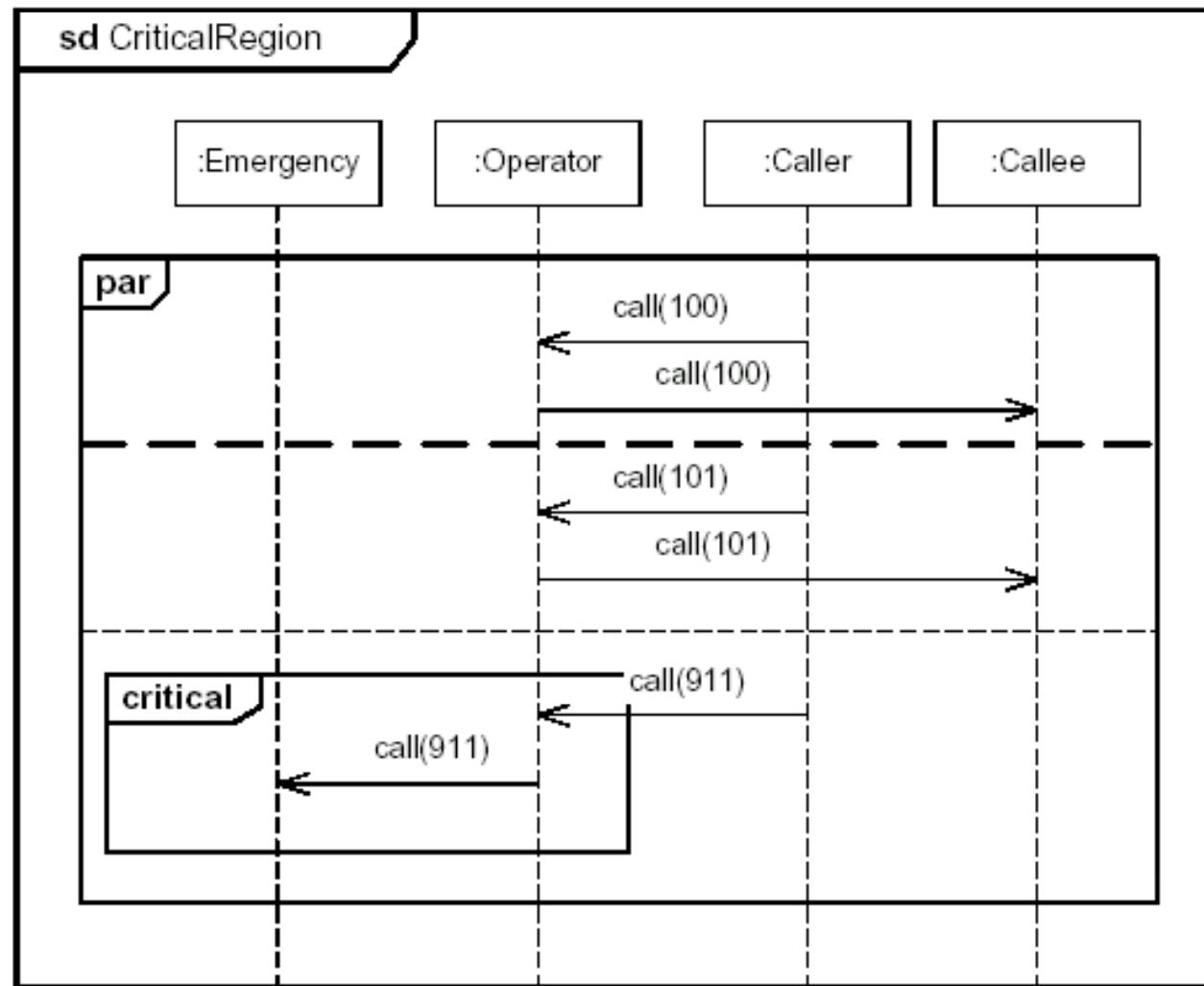
Diagramas de Secuencia

Operadores sobre interacciones. Bucle.



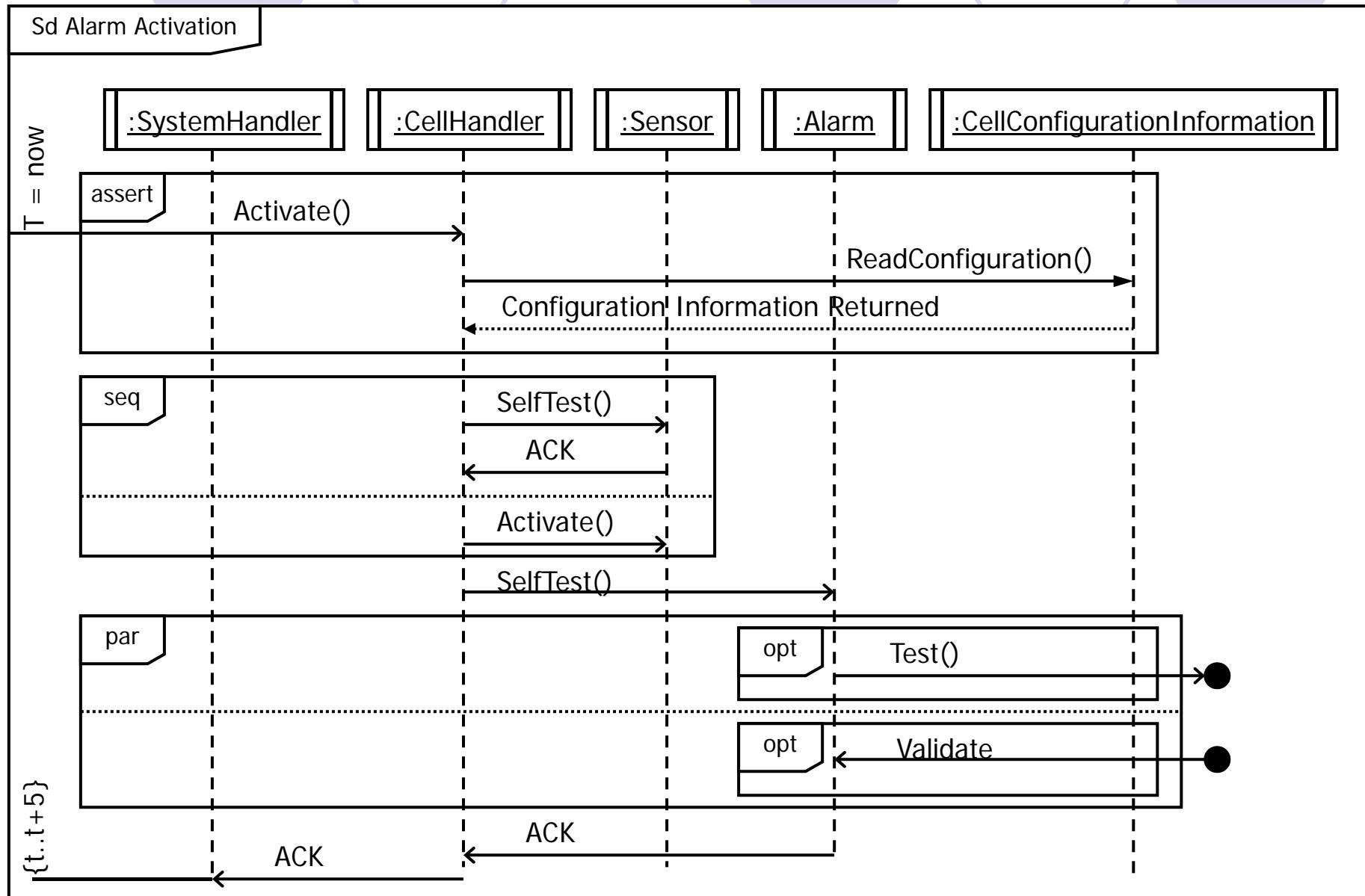
Diagramas de Secuencia

Operadores sobre interacciones. Región Crítica.



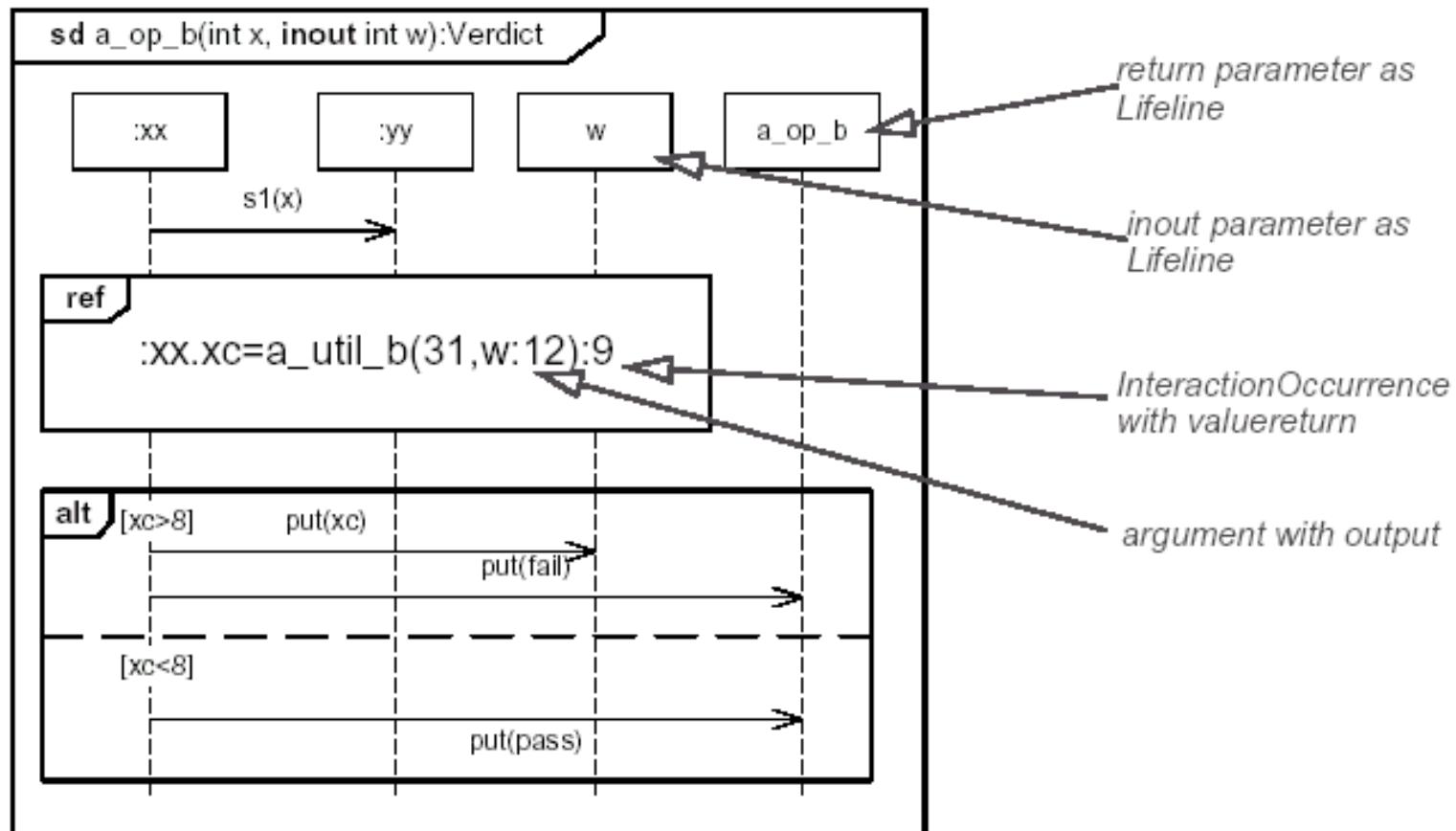
Diagramas de Secuencia

Ejemplo



Diagramas de Secuencia

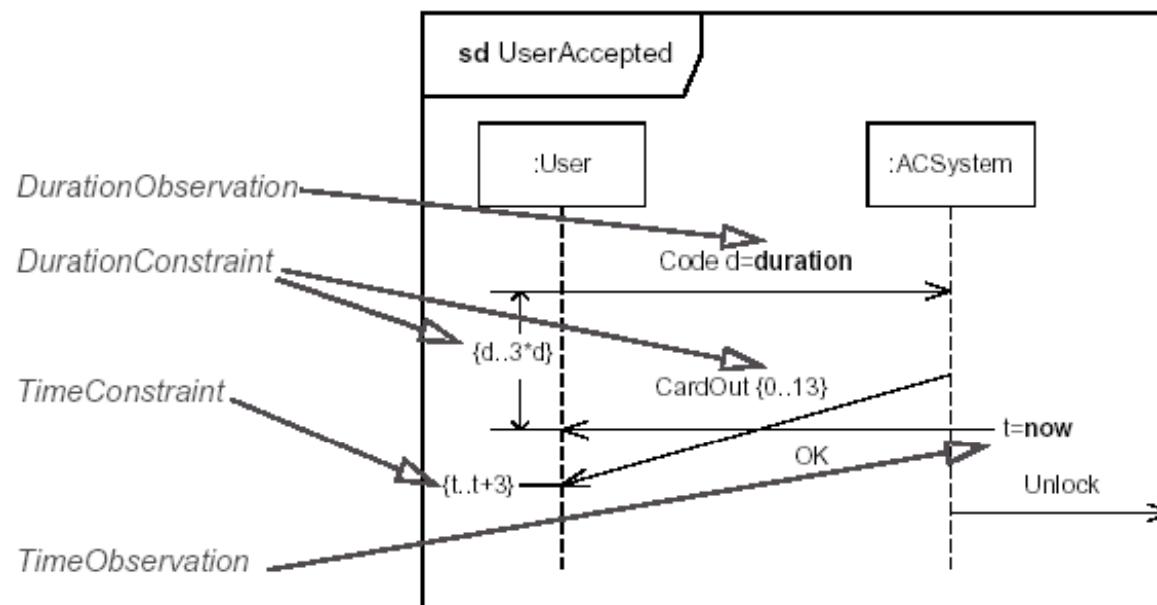
Retorno de Valores



Diagramas de Secuencia

Tiempo

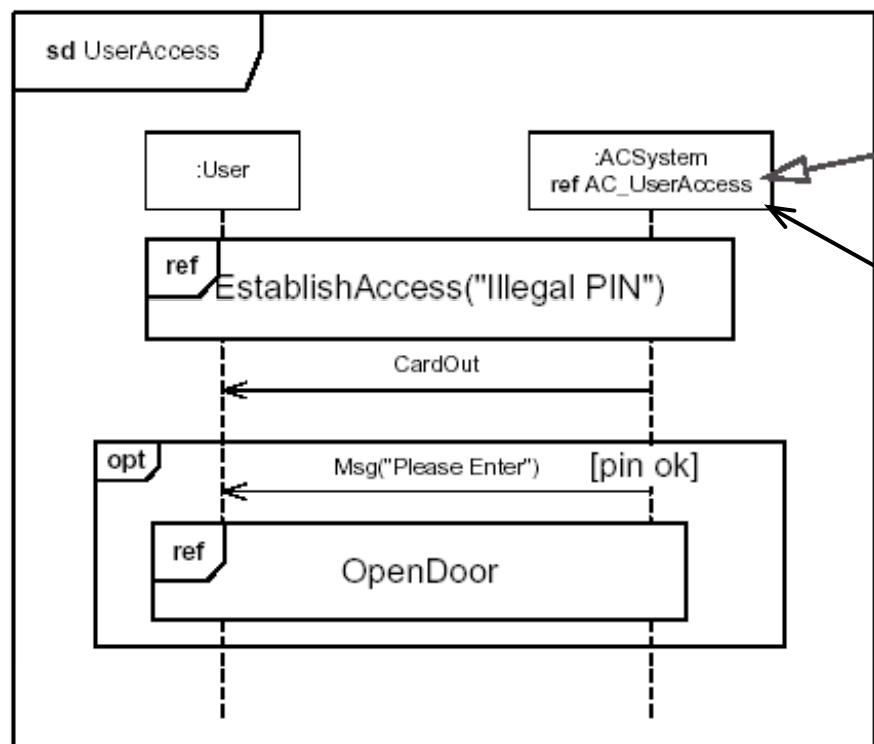
- Restricciones temporales (duración)



- Duración de mensajes y señales (**duration**)
- Intervalos de tiempo ($\{t..t+3\}$) y restricciones temporales.
- Observaciones temporales (tiempo actual, **now**)

Diagramas de Secuencia

Descomposición en partes

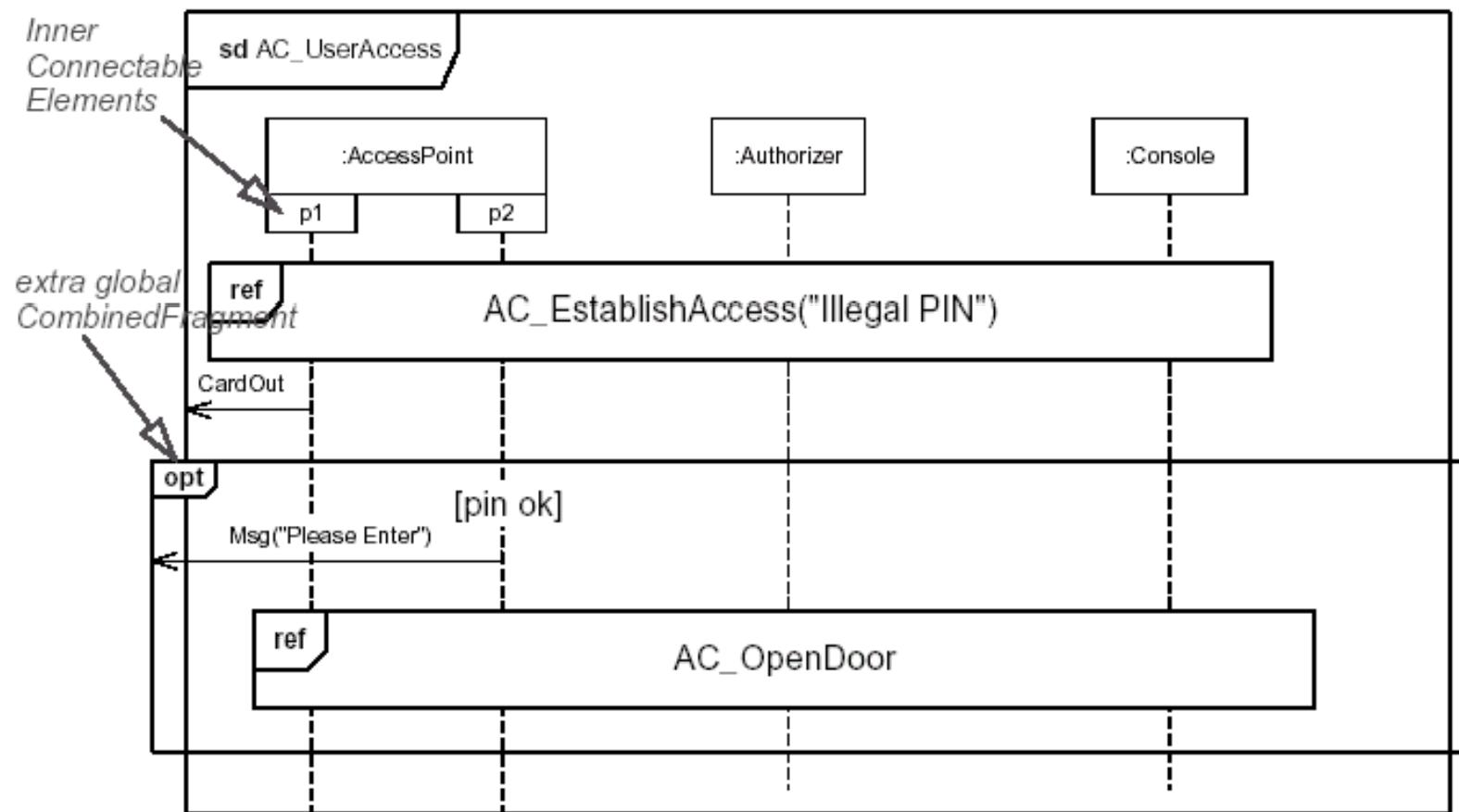


Part decomposition

La estructura interna de ACSystem tiene una interacción “AC_UserAccess” que se invoca en este fragmento.

Diagramas de Secuencia

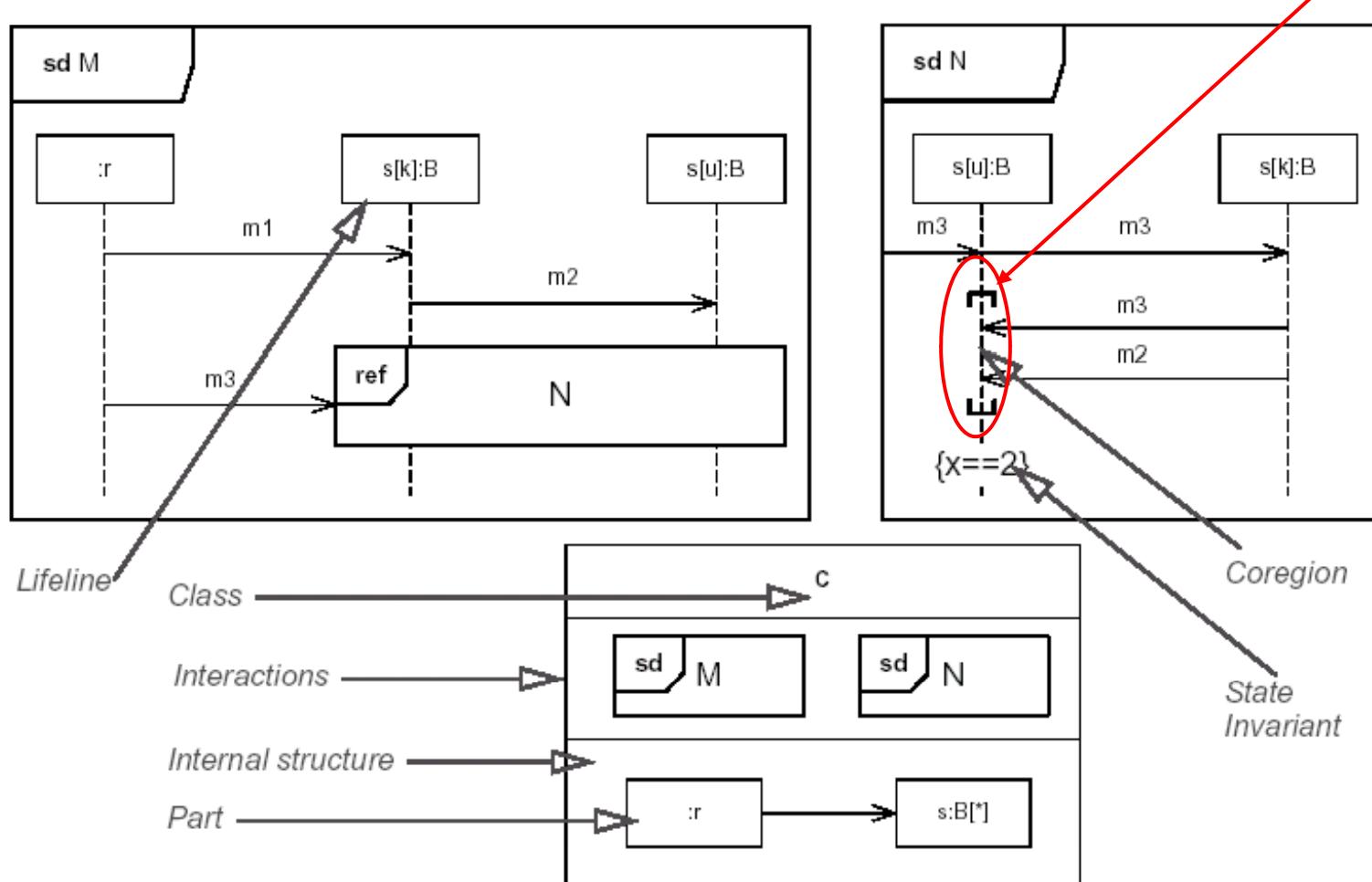
Descomposición en partes



Diagramas de Secuencia

co-región

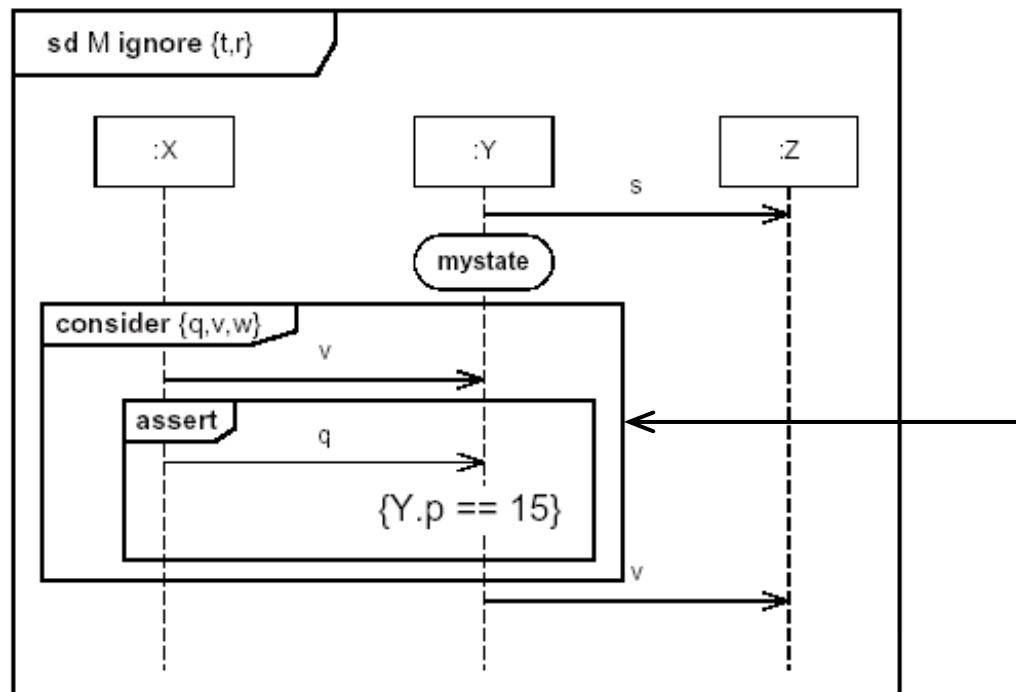
Coregión: $s[u]$ recibe los mensajes en cualquier orden.



Diagramas de Secuencia

Invariantes

- Invariantes.

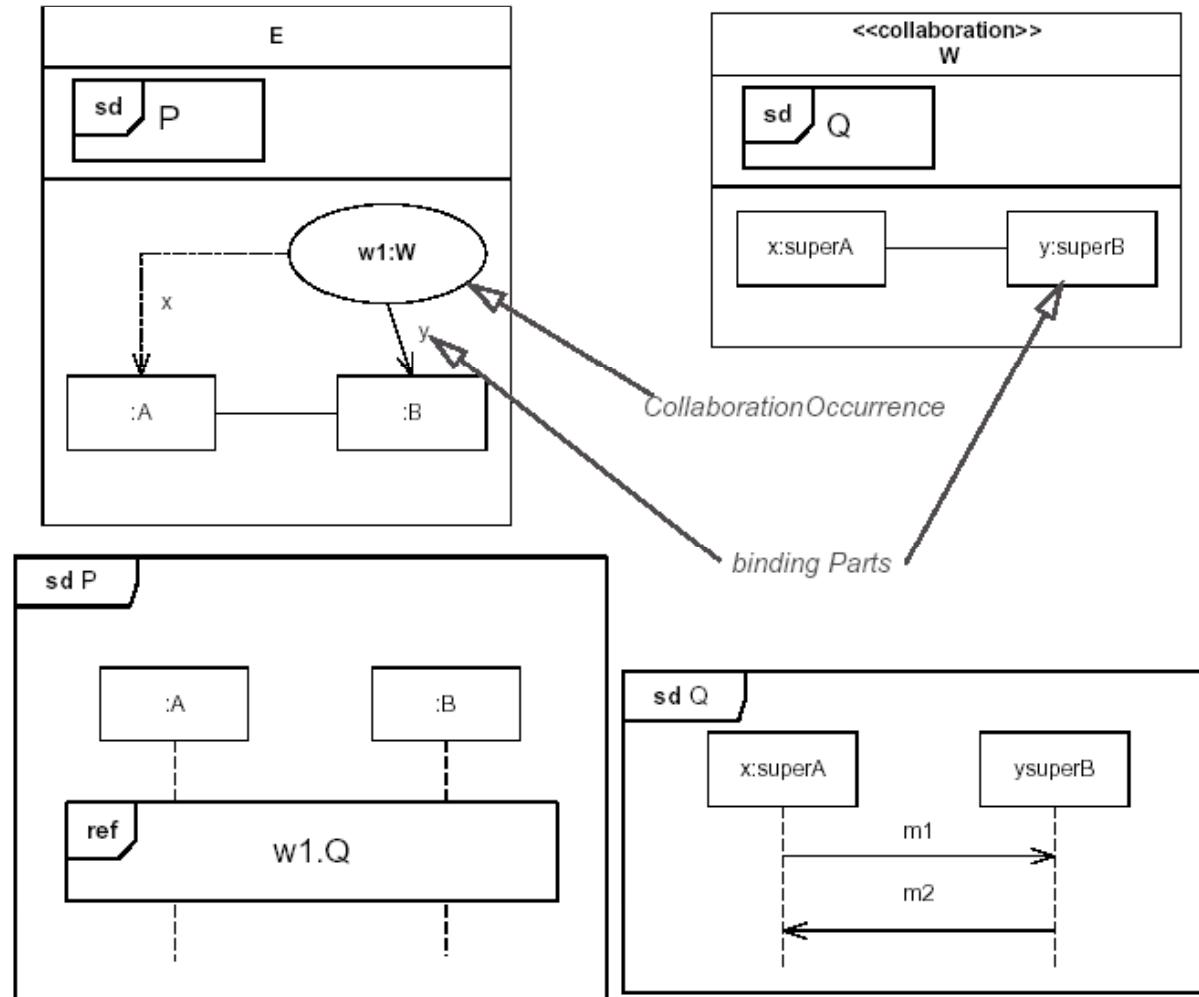


En este momento la traza
 $<v\ w\ q>$ no es válida

- Se comprueban justo antes de la ocurrencia del próximo evento (puede haber mensajes no mostrados en el diagrama).

Diagramas de Secuencia

Binding

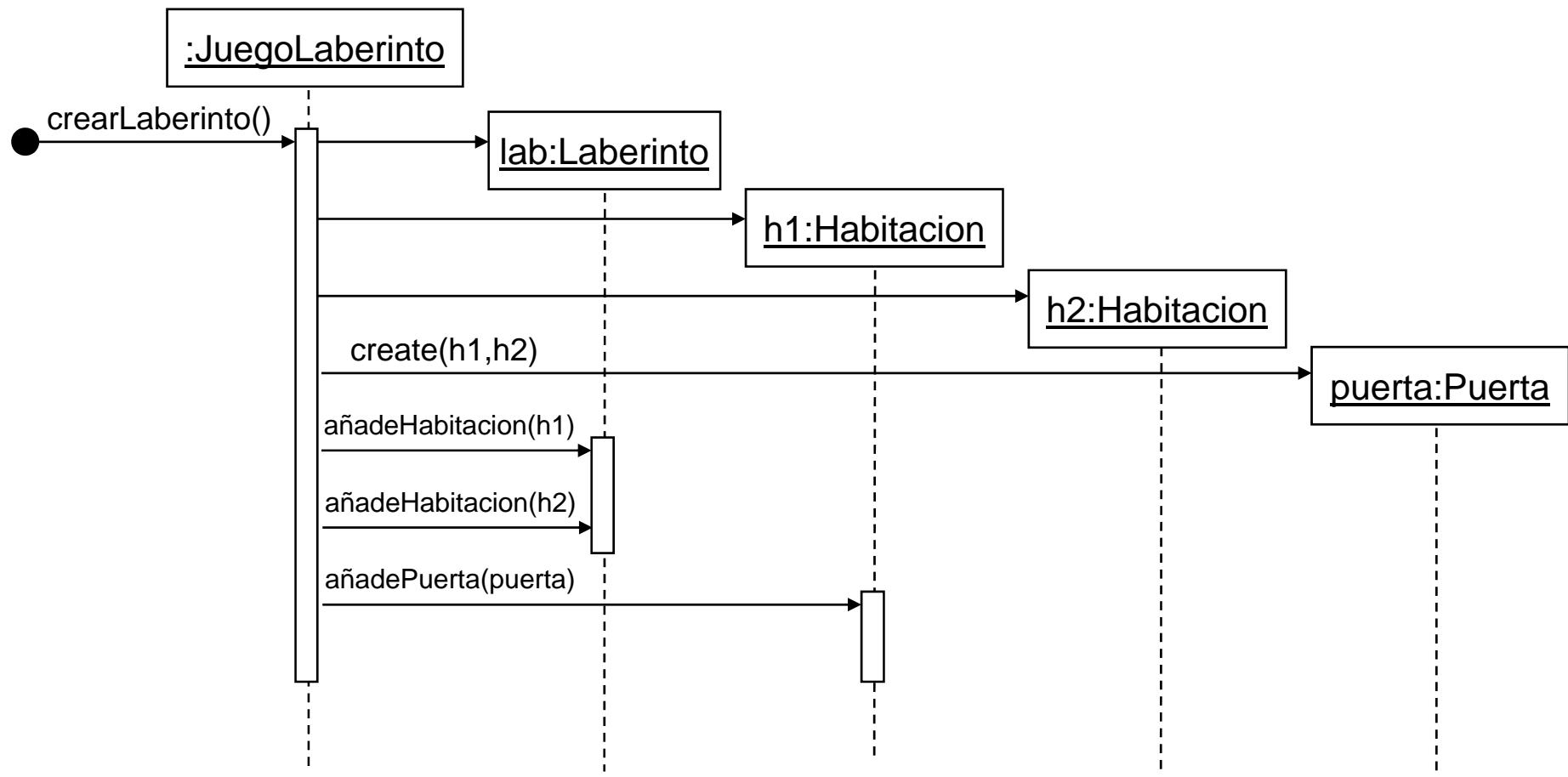


Ejercicio

Especificar el diagrama de secuencia de la operación
“crearLaberinto”

```
public class JuegoLaberinto {  
    public Laberinto crearLaberinto () {  
        Laberinto lab = new Laberinto();  
        Habitacion h1 = new Habitacion();  
        Habitacion h2 = new Habitacion();  
        Puerta puerta = new Puerta(h1, h2);  
        lab.añadeHabitacion(h1);  
        lab.añadeHabitacion(h2);  
        h1.añadePuerta(puerta);  
        return lab;  
    }  
}
```

Solución

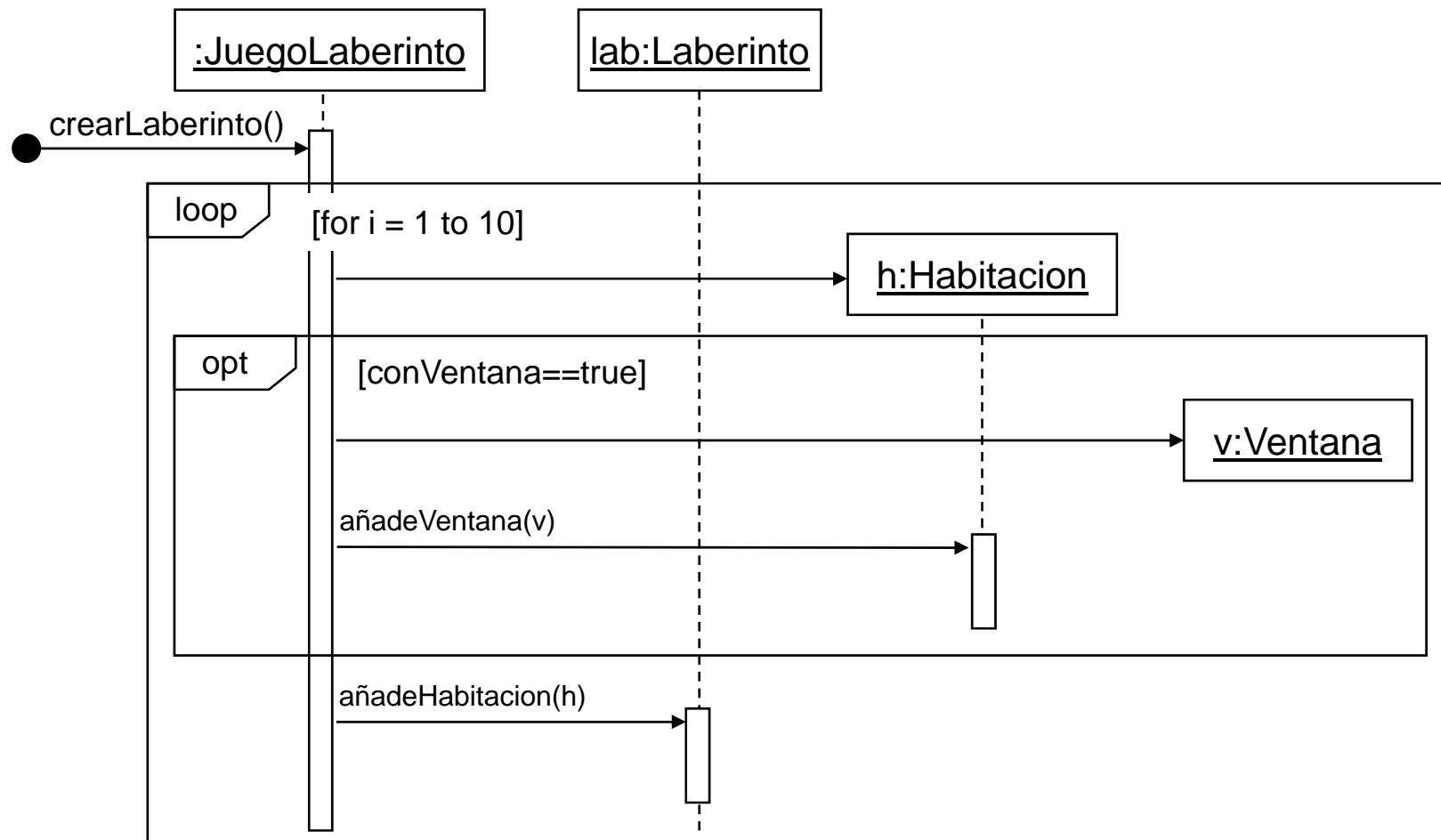


Ejercicio

Especificar el diagrama de secuencia de la operación
“crearLaberinto”

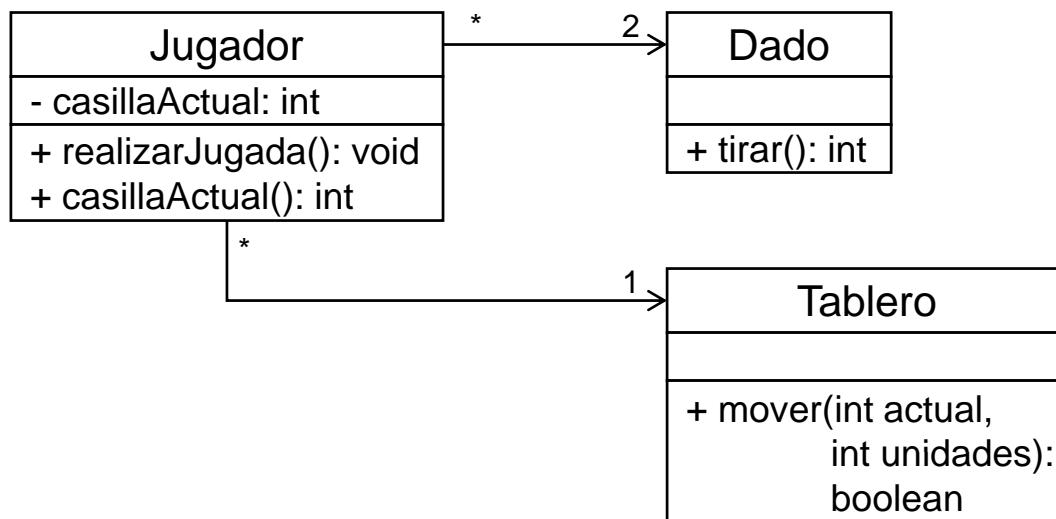
```
public class JuegoLaberinto {  
    private Laberinto lab;  
    private boolean conVentana;  
  
    public JuegoLaberinto() {  
        lab = new Laberinto();  
        conVentana = true;  
    }  
  
    public void crearLaberinto () {  
        Habitacion h;  
        for (int i=0; i<10; i++) {  
            h = new Habitacion();  
            if (conVentana == true)  
                h.añadeVentana(new Ventana());  
            lab.añadeHabitacion(h);  
        }  
    }  
}
```

Solución

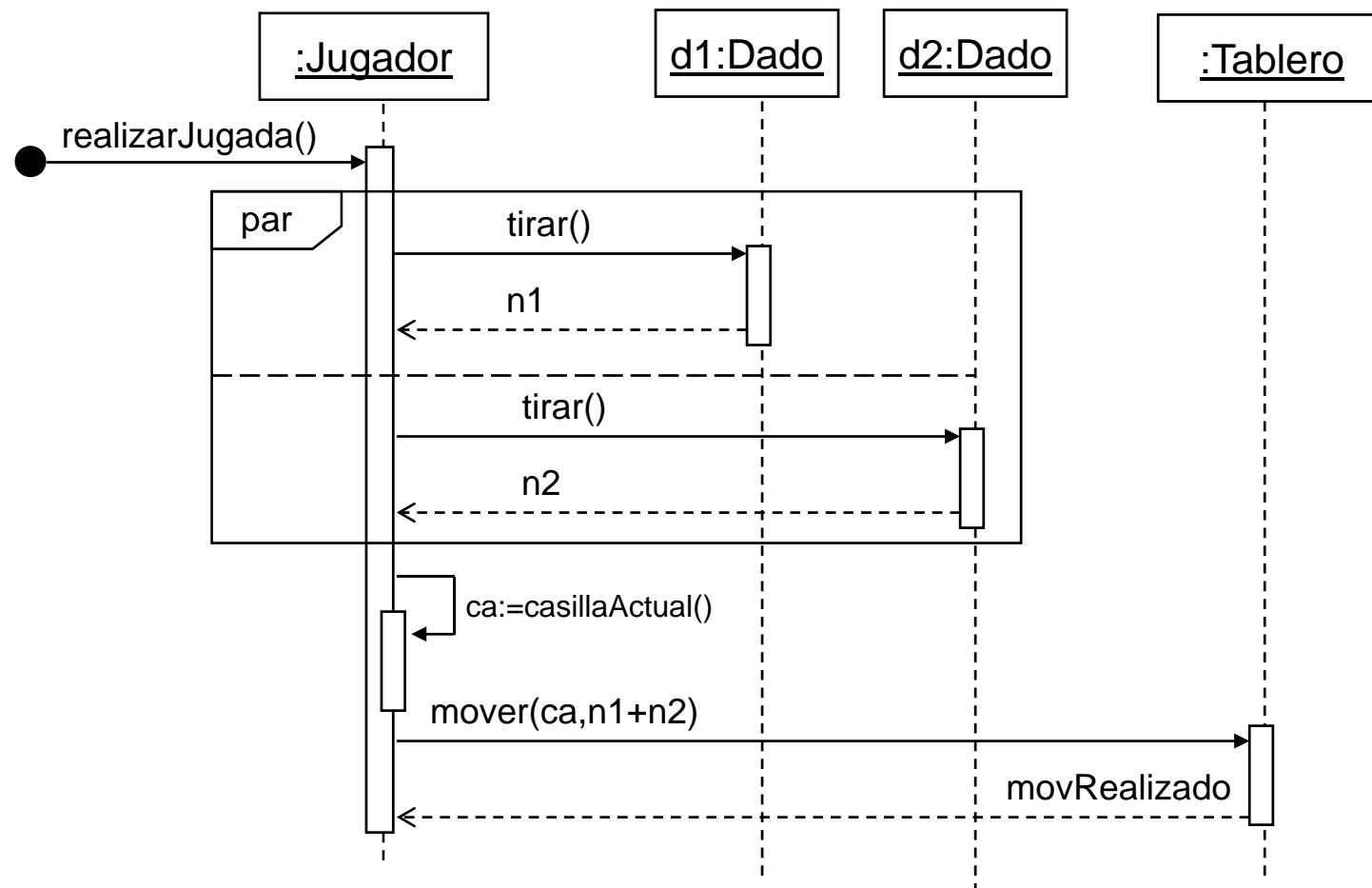


Ejercicio

Especificar el diagrama de secuencia de la operación
“realizarJugada” definida en la clase Jugador, para el juego del
parchís



Solución

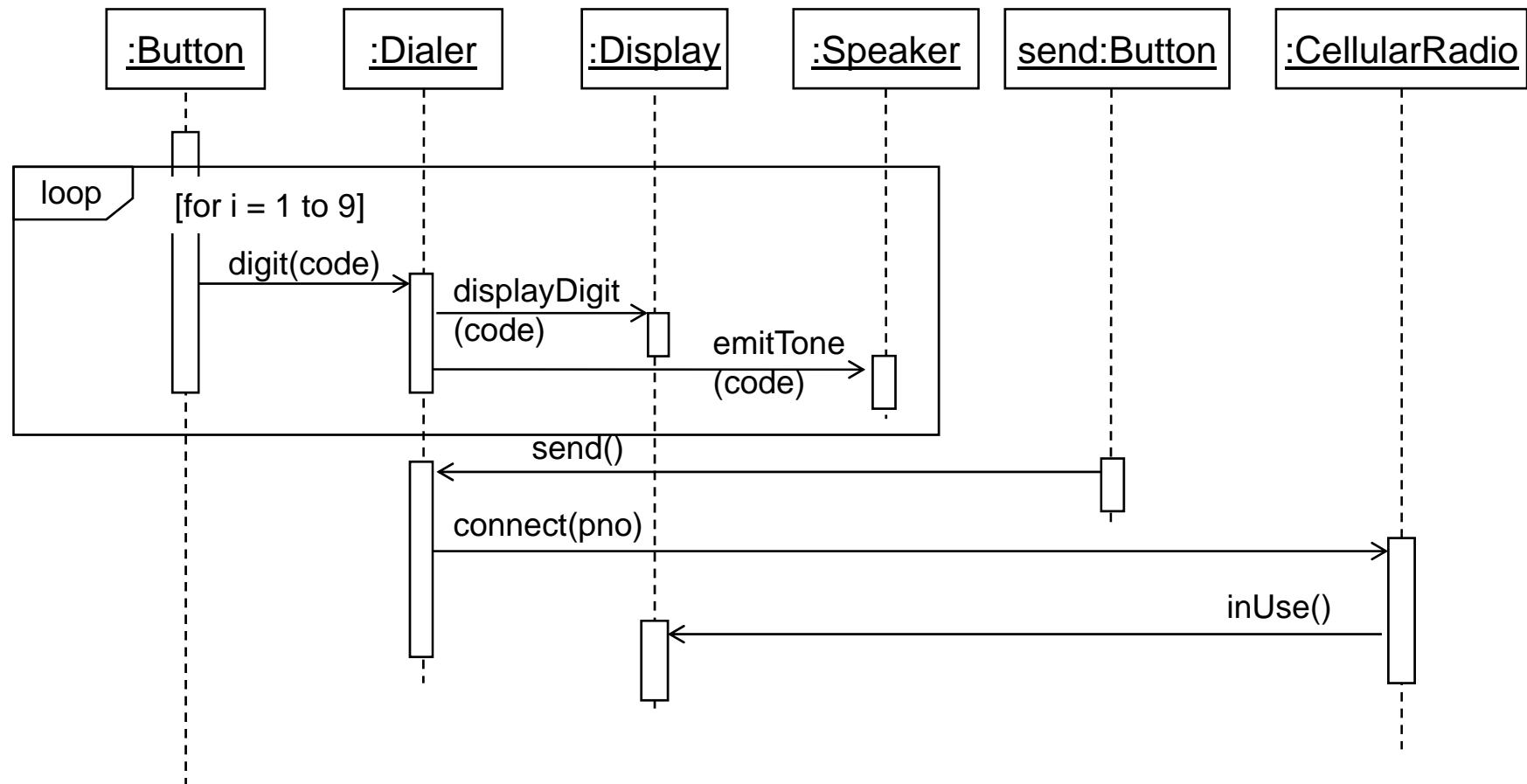


Ejercicio

Identificar las clases relevantes y realizar el diagrama de secuencia para el siguiente caso de uso, que corresponde a la realización de una llamada desde un teléfono móvil.

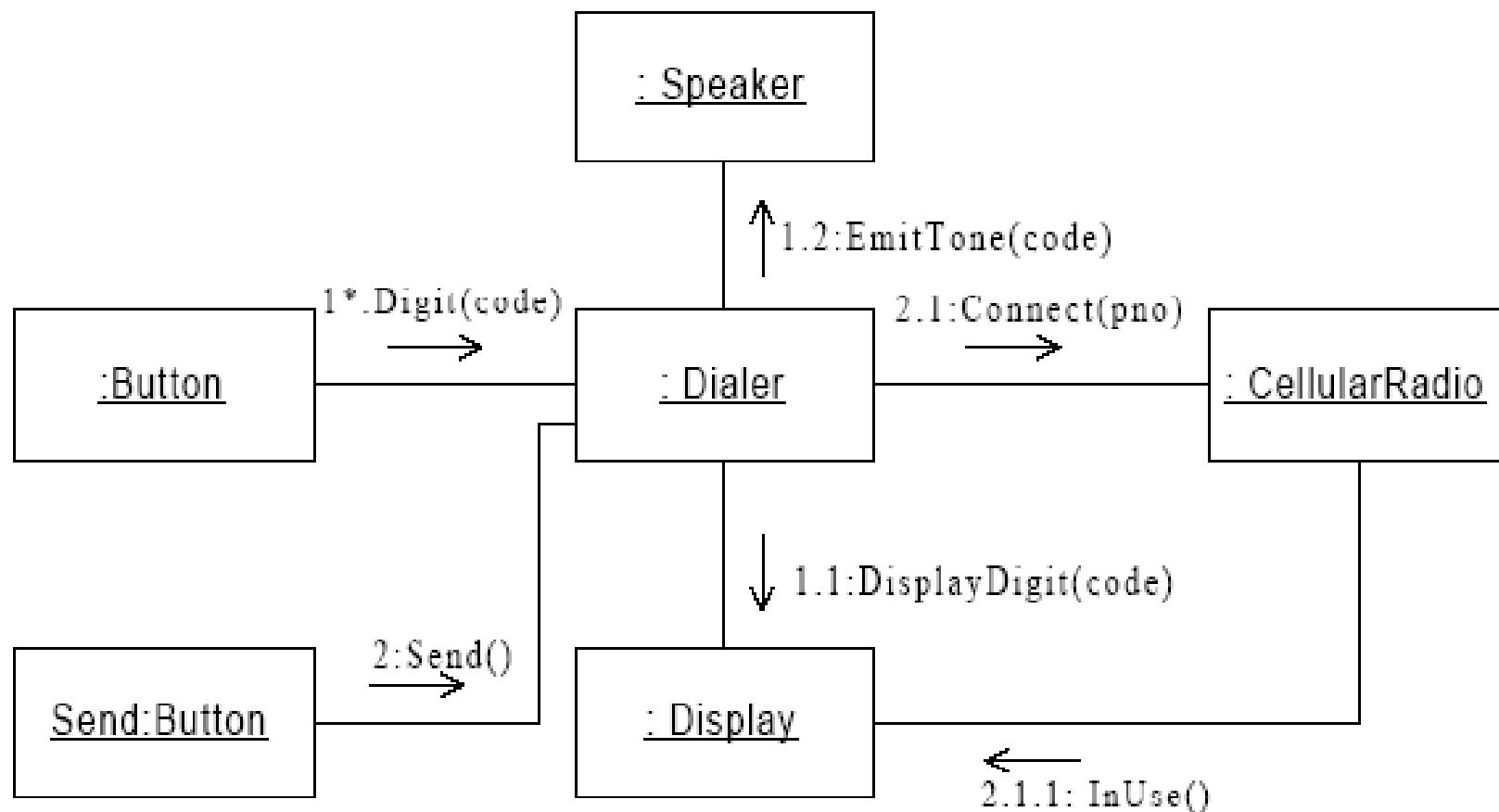
- El usuario pulsa los dígitos del número de teléfono
 - Para cada dígito
 - la pantalla se actualiza para añadir el dígito marcado
 - se emite un tono por el receptor
- El usuario pulsa el botón “Enviar”
- El indicador “en uso” se ilumina en pantalla
- El móvil establece conexión con la red
- Los dígitos acumulados se mandan a la red
- Se establece la conexión con el número marcado

Solución



¿Diagrama de colaboración equivalente?

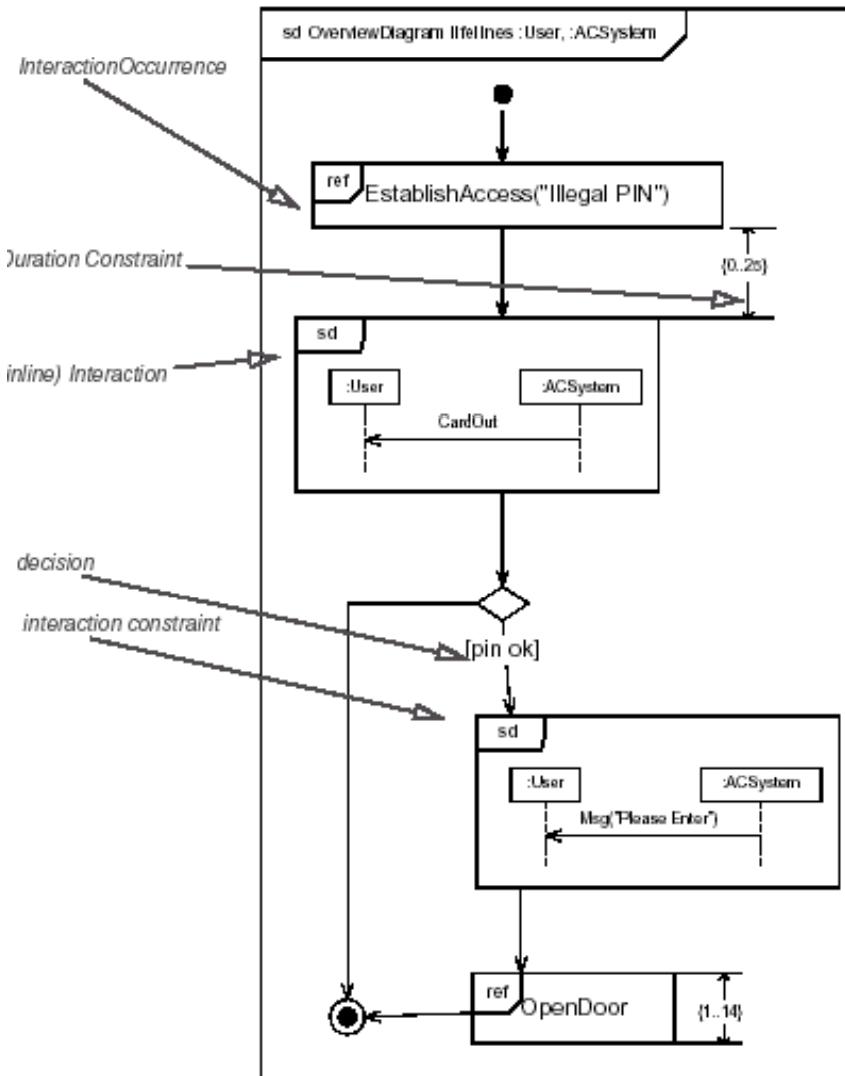
Solución



Visión de Conjunto de la Interacción

- Define las interacciones a través de una variante de los diagramas de actividad.
- Visión general del flujo de control.
- Usa elementos de los diagramas de actividad para especificar:
 - Alternativas entre interacciones.
 - Paralelismo de interacciones.
 - Bucles de interacciones.

Visión de Conjunto de la Interacción



Tiempo

- Muestran interacciones donde es importante razonar sobre el tiempo.
- Representa condiciones que cambian en una o varias líneas de vida, en un eje lineal de tiempo.
- Cambios en el estado de un objeto con el tiempo en respuesta a eventos.

Tiempo

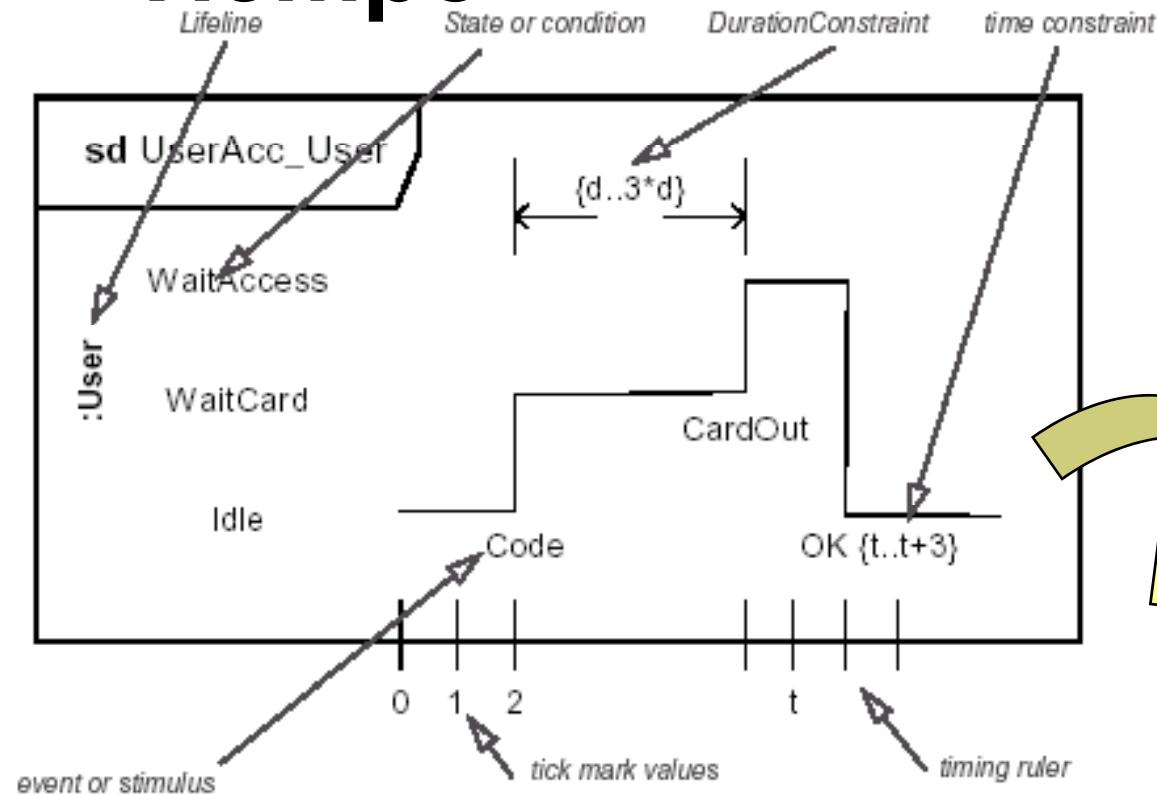
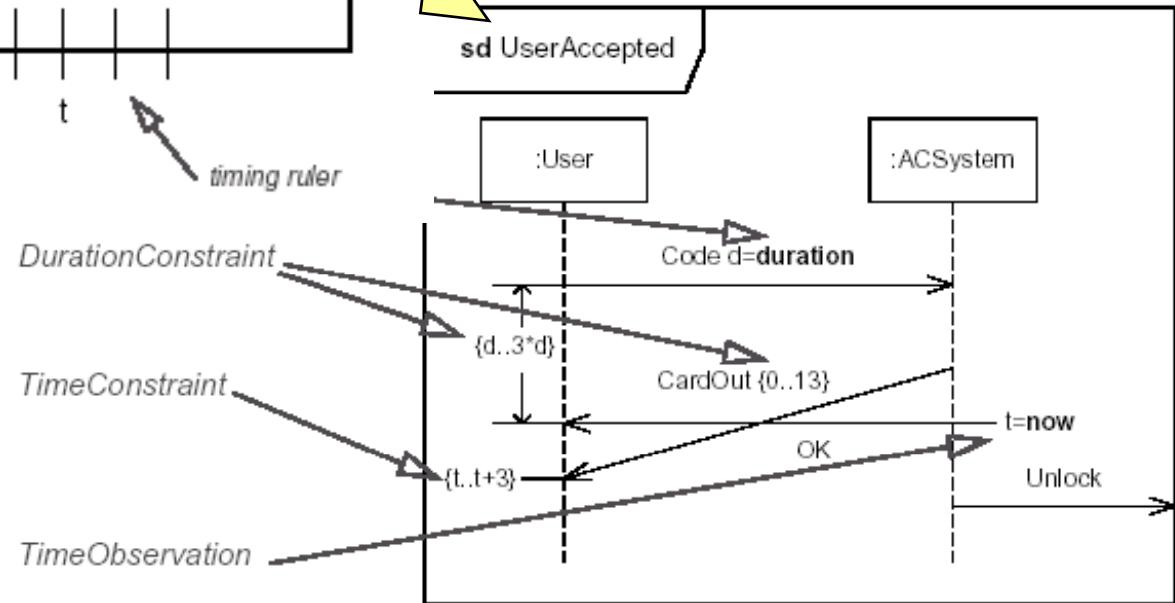
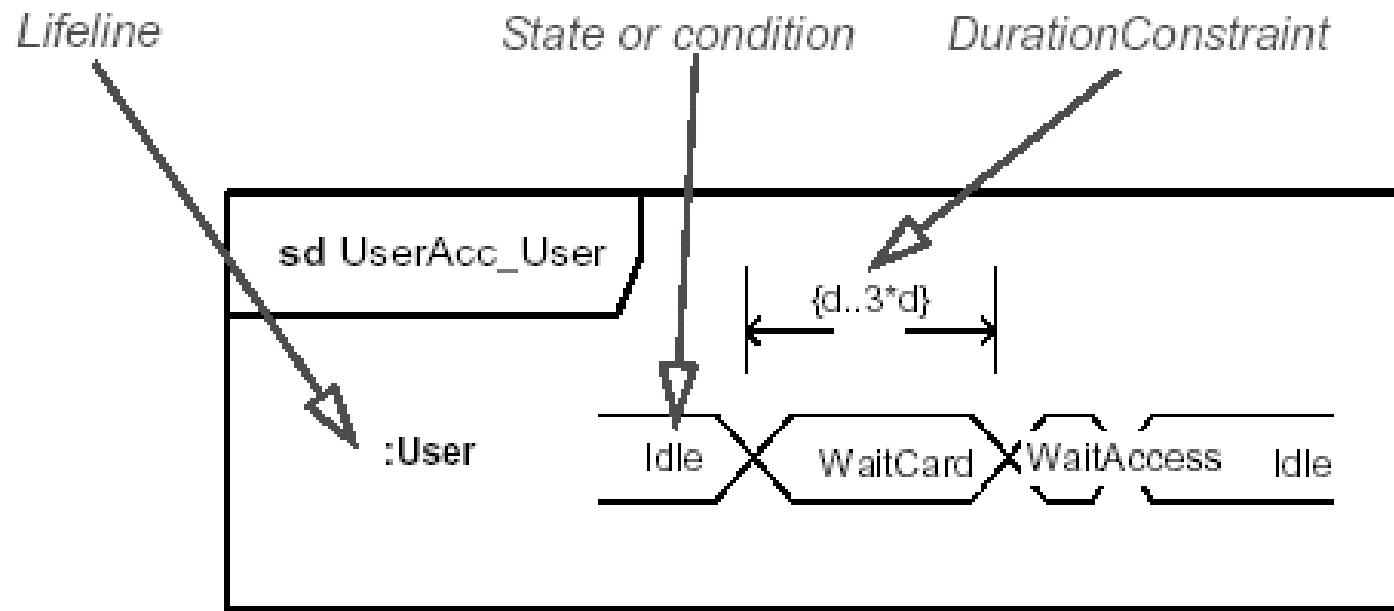


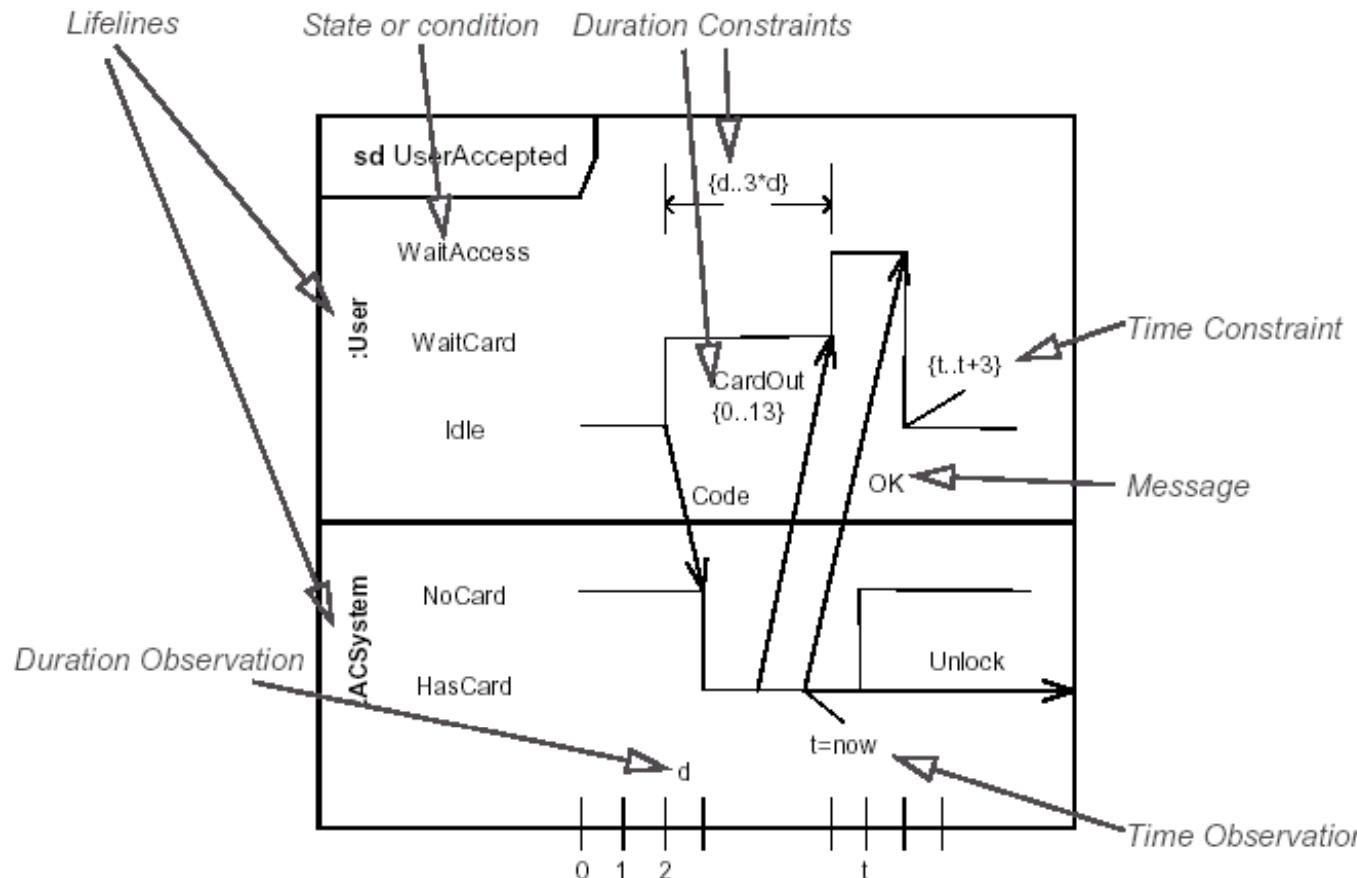
Diagrama temporal correspondiente a la interacción



Tiempo



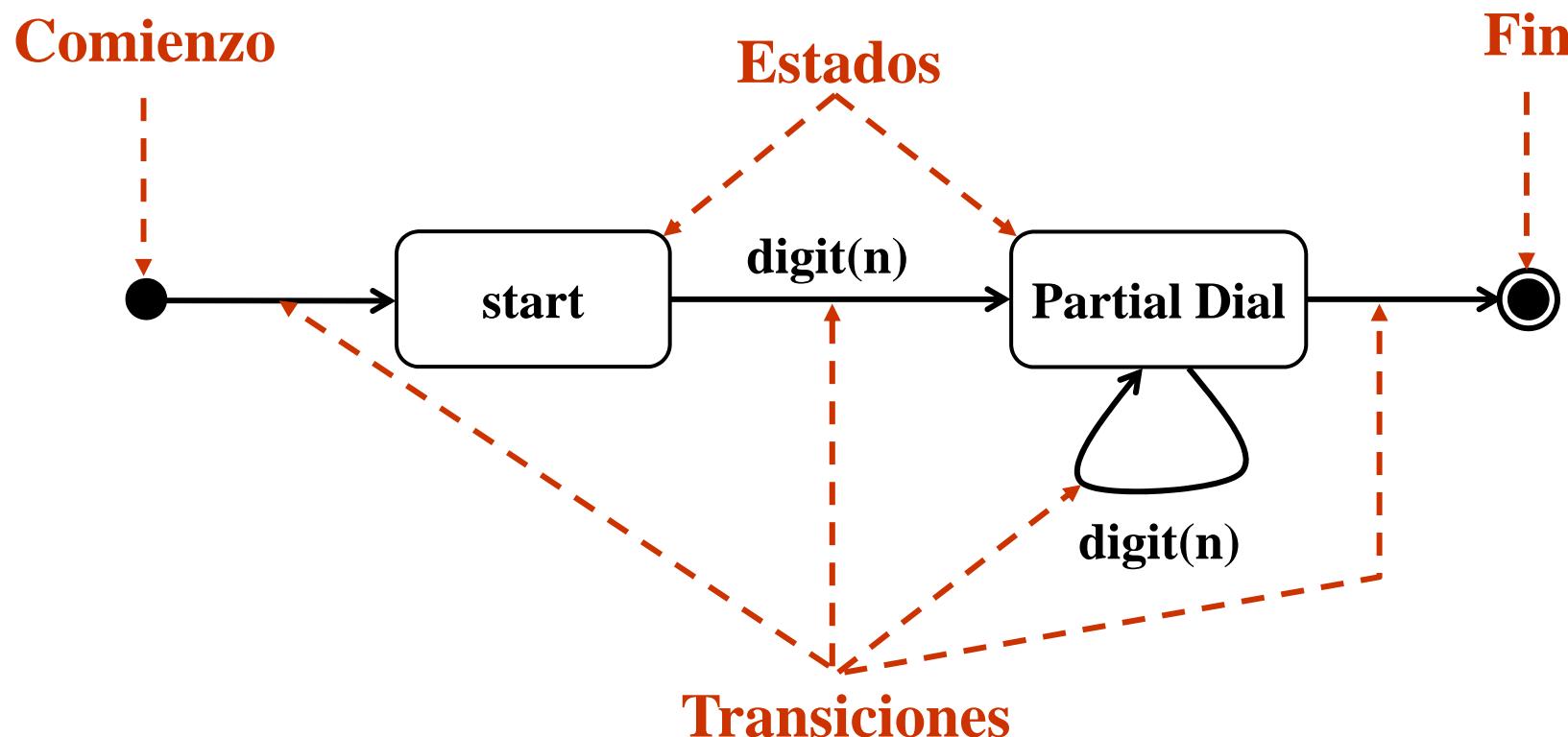
Tiempo



Máquinas de Estados

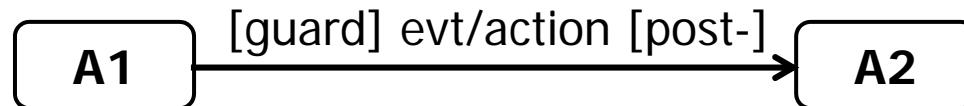
- “Statecharts” [Harel]
- Representan el comportamiento de entidades (p.e. instancias de clases).
- Especifican reacción ante eventos
- Describen posibles secuencias de estados y acciones por las que pueden pasar las entidades.
- De comportamiento vs. de protocolo.

Máquinas de Estados



Máquinas de Estados

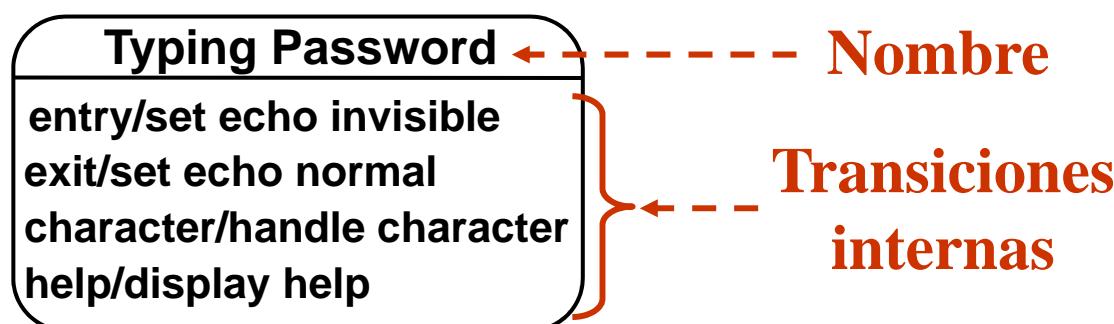
- Un transición puede tener:
 - Evento.
 - Eventos temporales: $tm(n)$
 - Acción.
 - pre-condiciones (guardas) y post- condiciones.



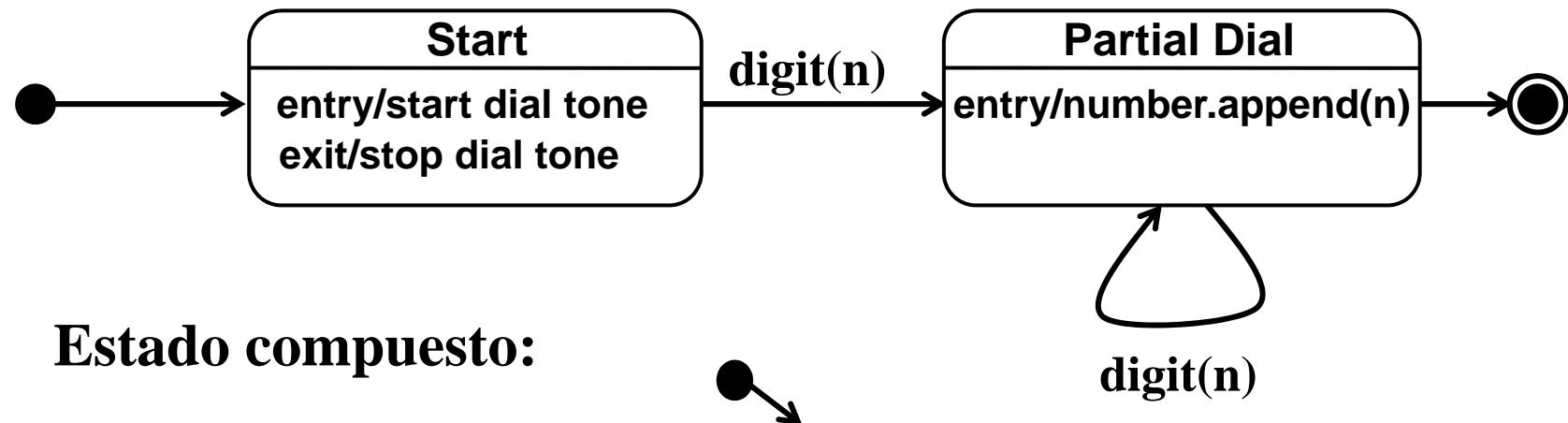
- Símbolos especiales para el envío y recepción de señales (normalmente usados en diagramas de actividad).

Máquinas de Estados

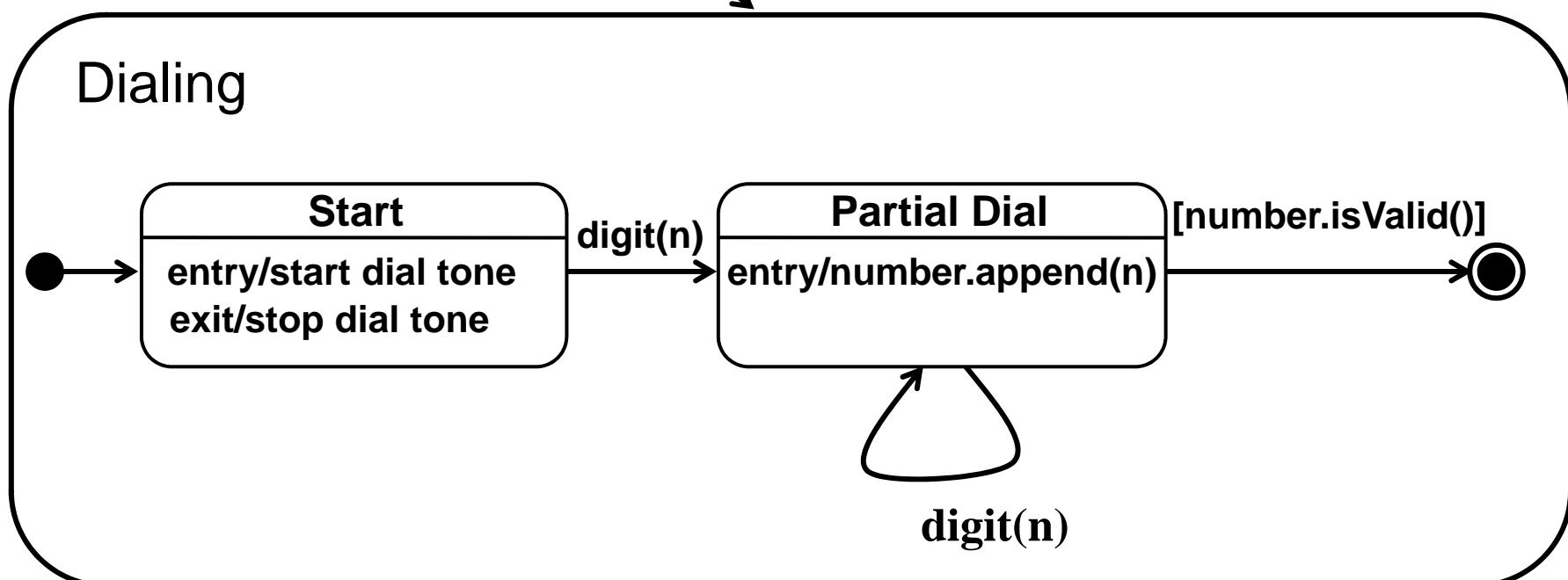
- Un estado tiene:
 - Nombre
 - Transiciones internas: lista de acciones ejecutadas en ese estado (entry/exit/do)
- Ejemplo:



Máquinas de Estados



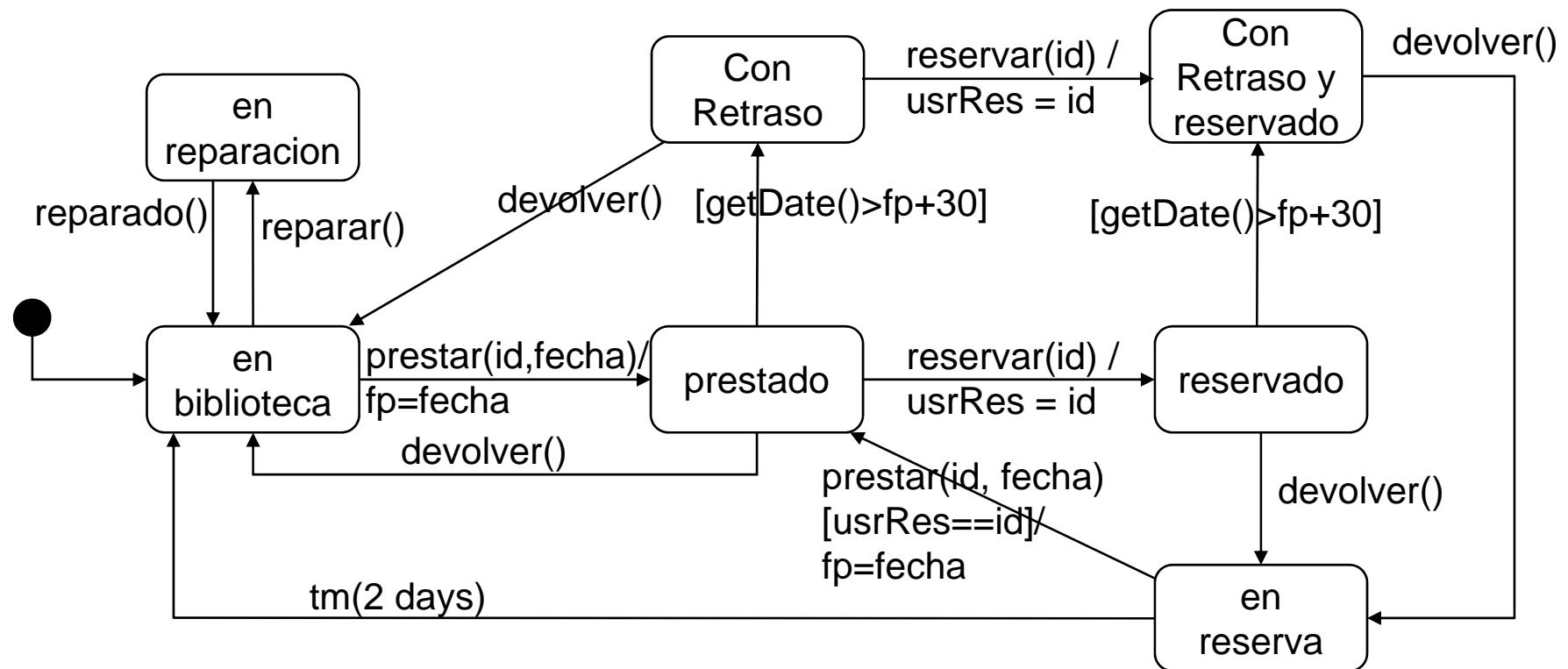
Estado compuesto:



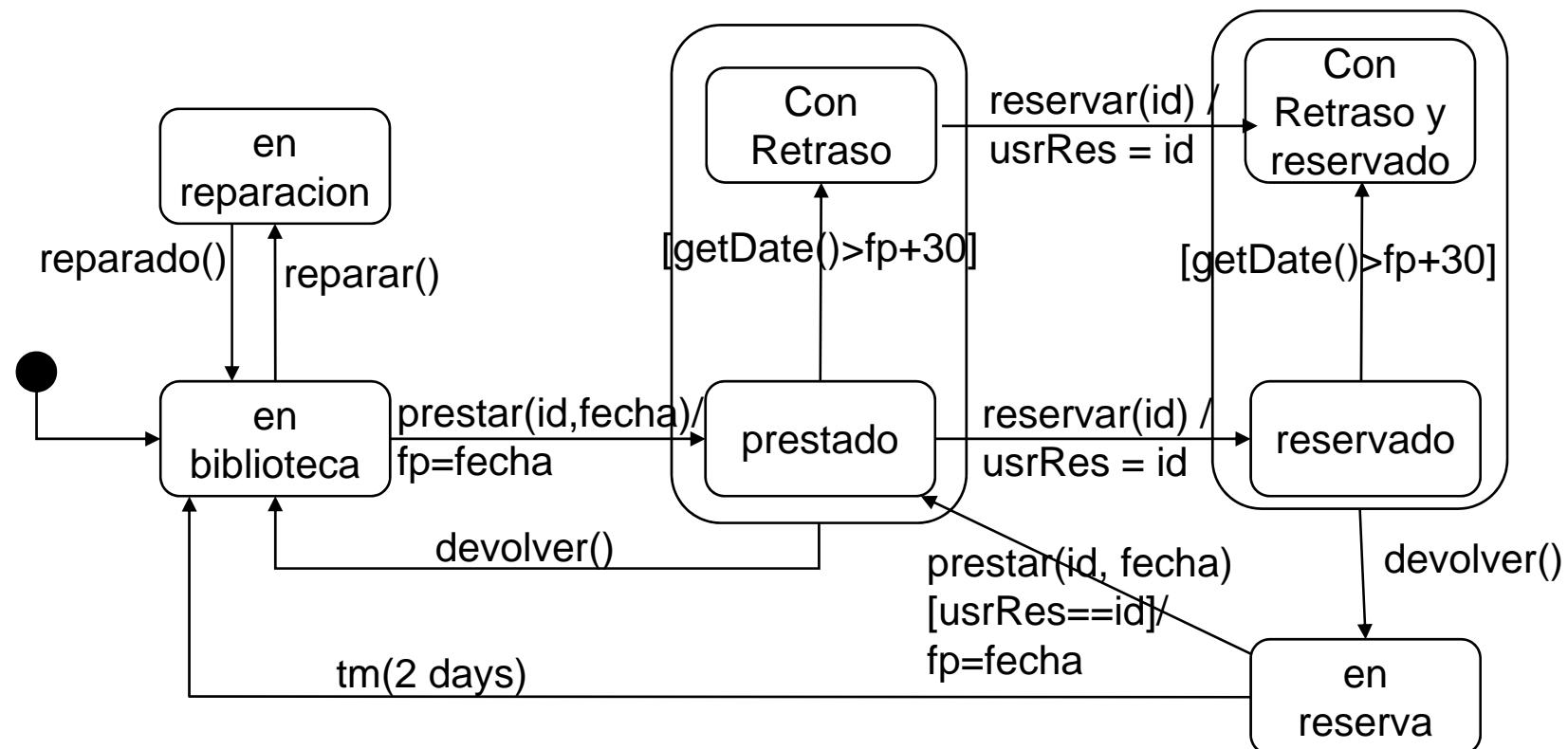
Ejercicio: Biblioteca

- Una biblioteca tiene copias de libros. Estos últimos se caracterizan por su nombre, tipo (novela, teatro, poesía, ensayo), editorial, año y autor.
- Los autores se caracterizan por su nombre, nacionalidad y fecha de nacimiento.
- Cada copia tiene un identificador, y puede estar en la biblioteca, prestada, reservada, con retraso o en reparación.
- Los lectores pueden tener un máximo de 3 libros en préstamo.
- Cada libro se presta un máximo de 30 días, por cada día de retraso, se impone una “multa” de dos días sin posibilidad de coger un libro.
- Realiza el diagrama de estados de la clase “copia”.

Solucion

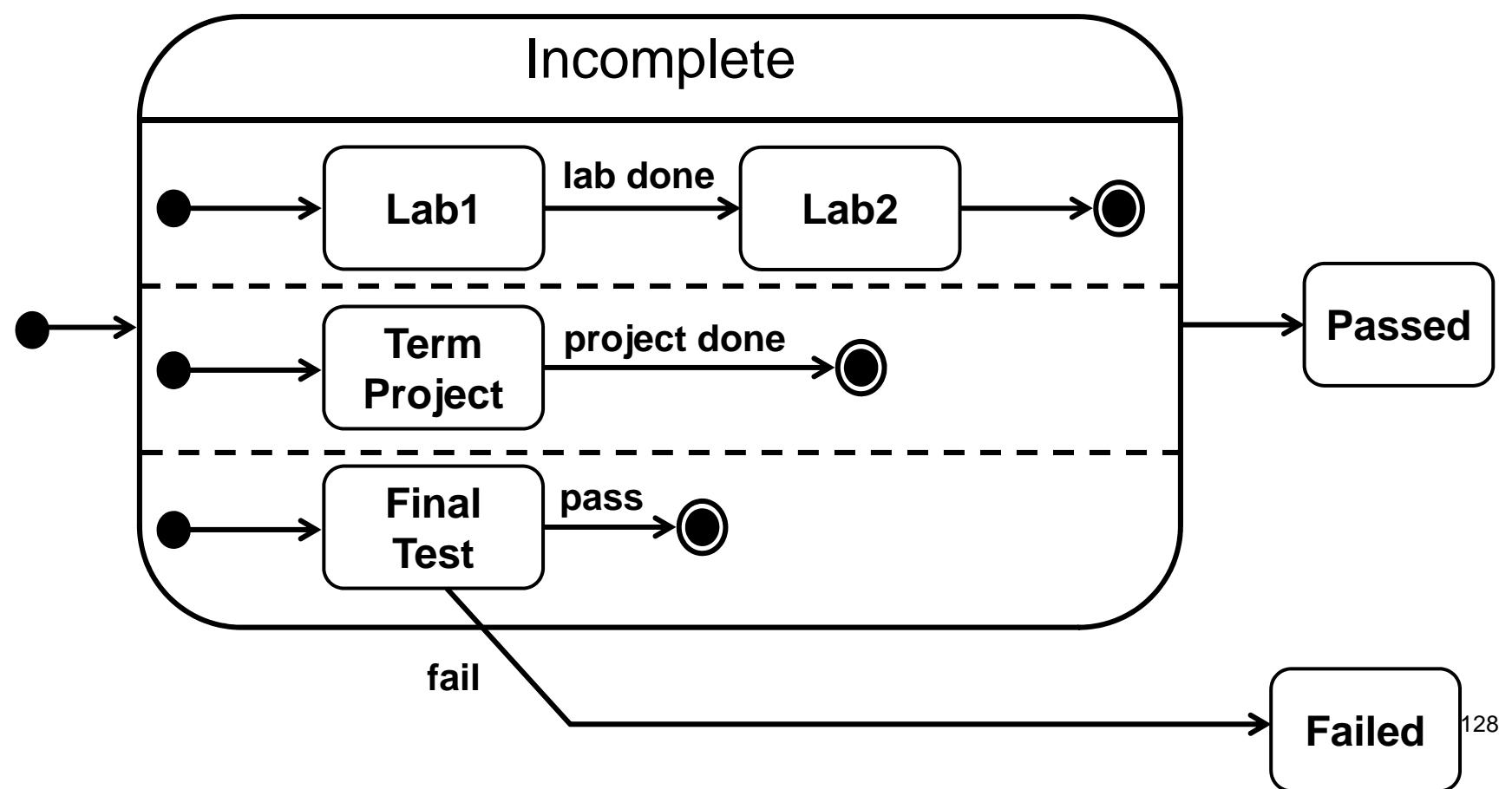


Solucion: Estados Jerárquicos



Máquinas de Estados

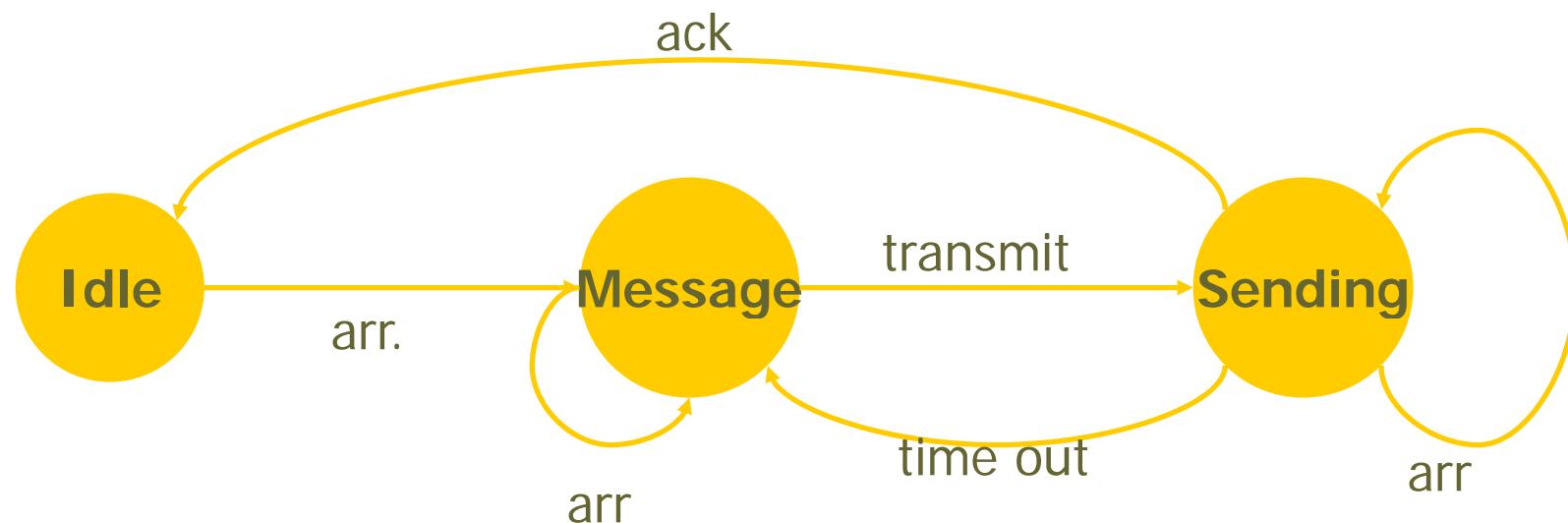
Componentes Ortogonales



Máquinas de Estados

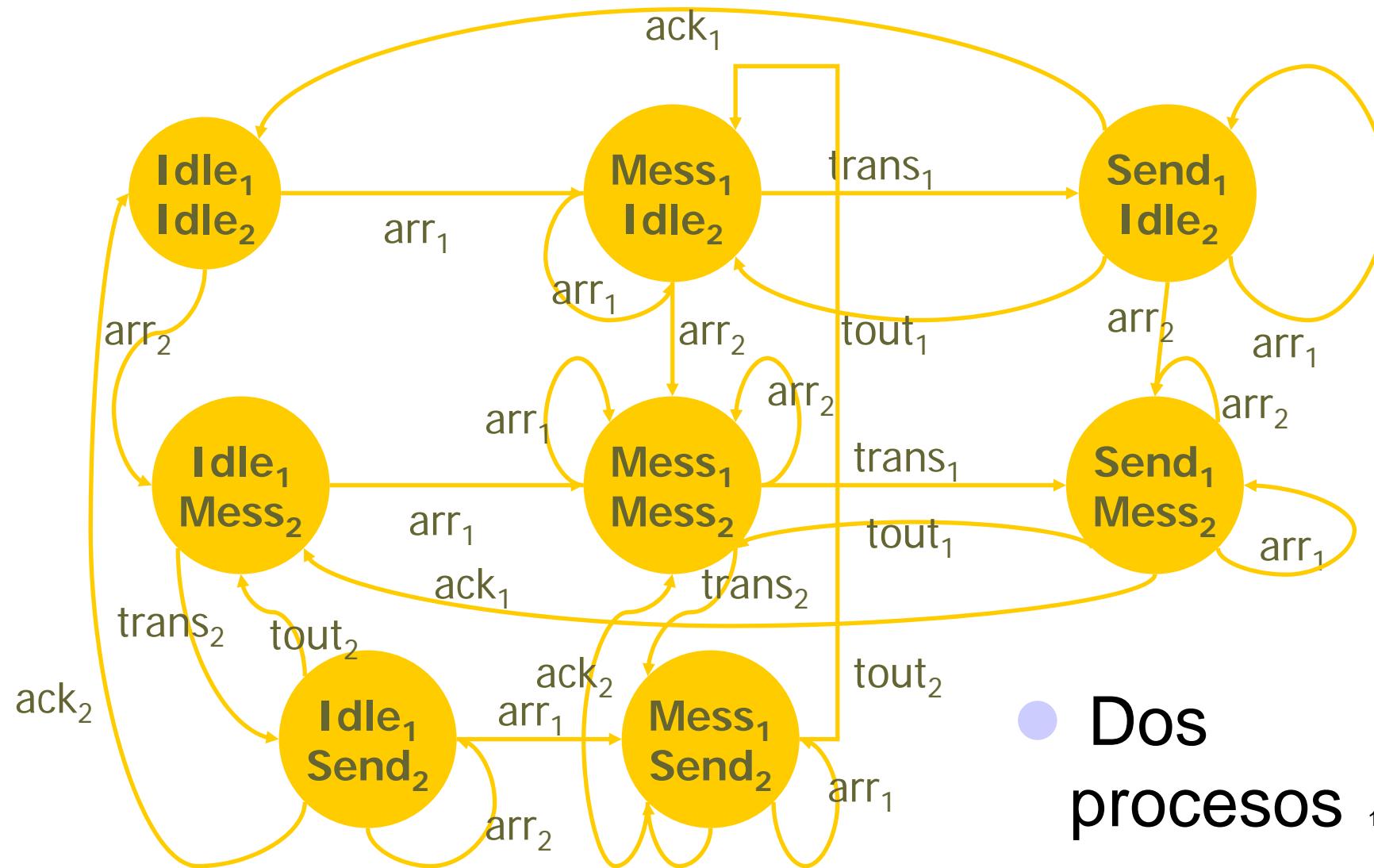
Componentes Ortogonales: Utilidad

- Modelo de un sistema formado por un proceso en red sin componentes ortogonales.



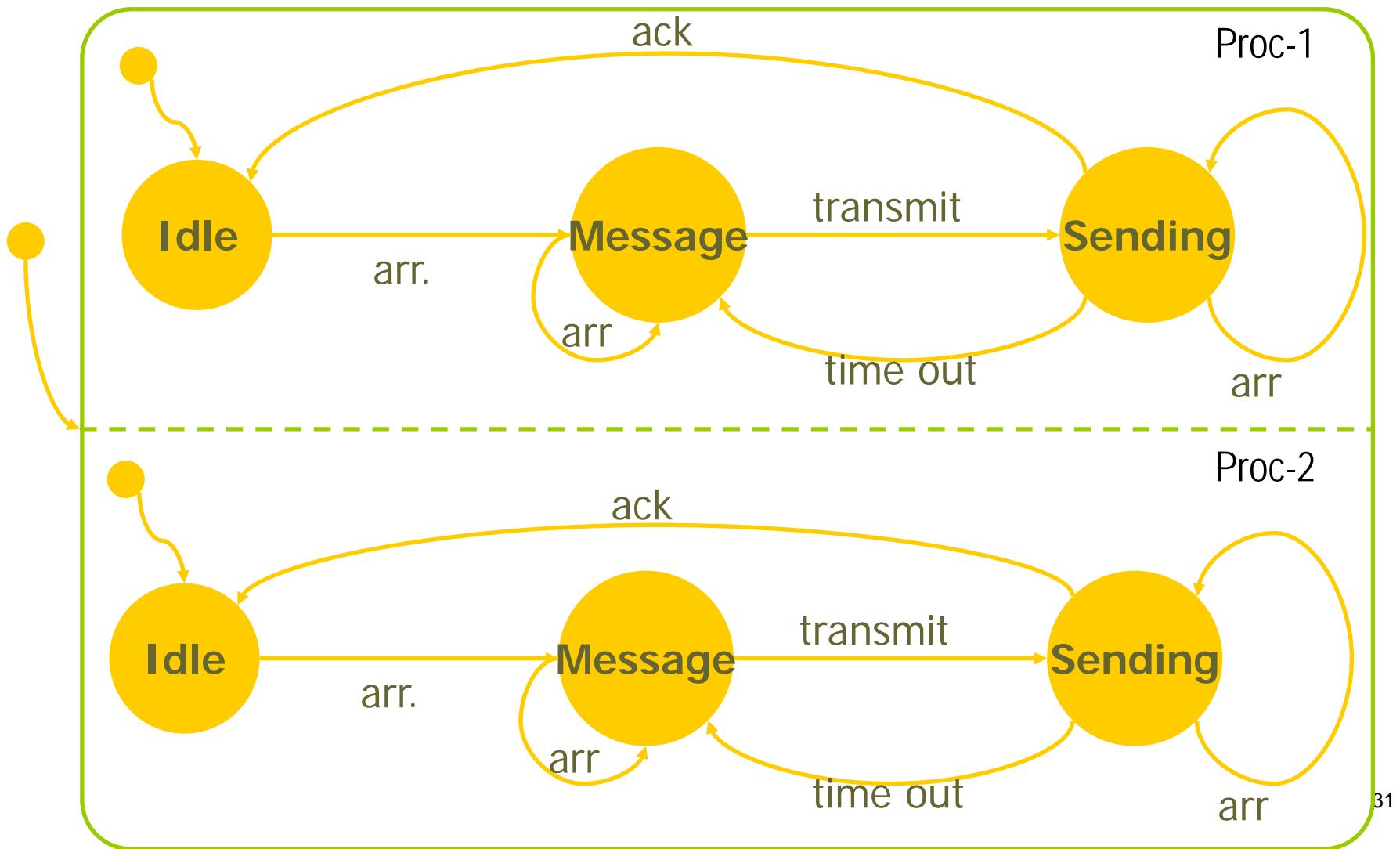
Máquinas de Estados

Componentes Ortogonales: Utilidad



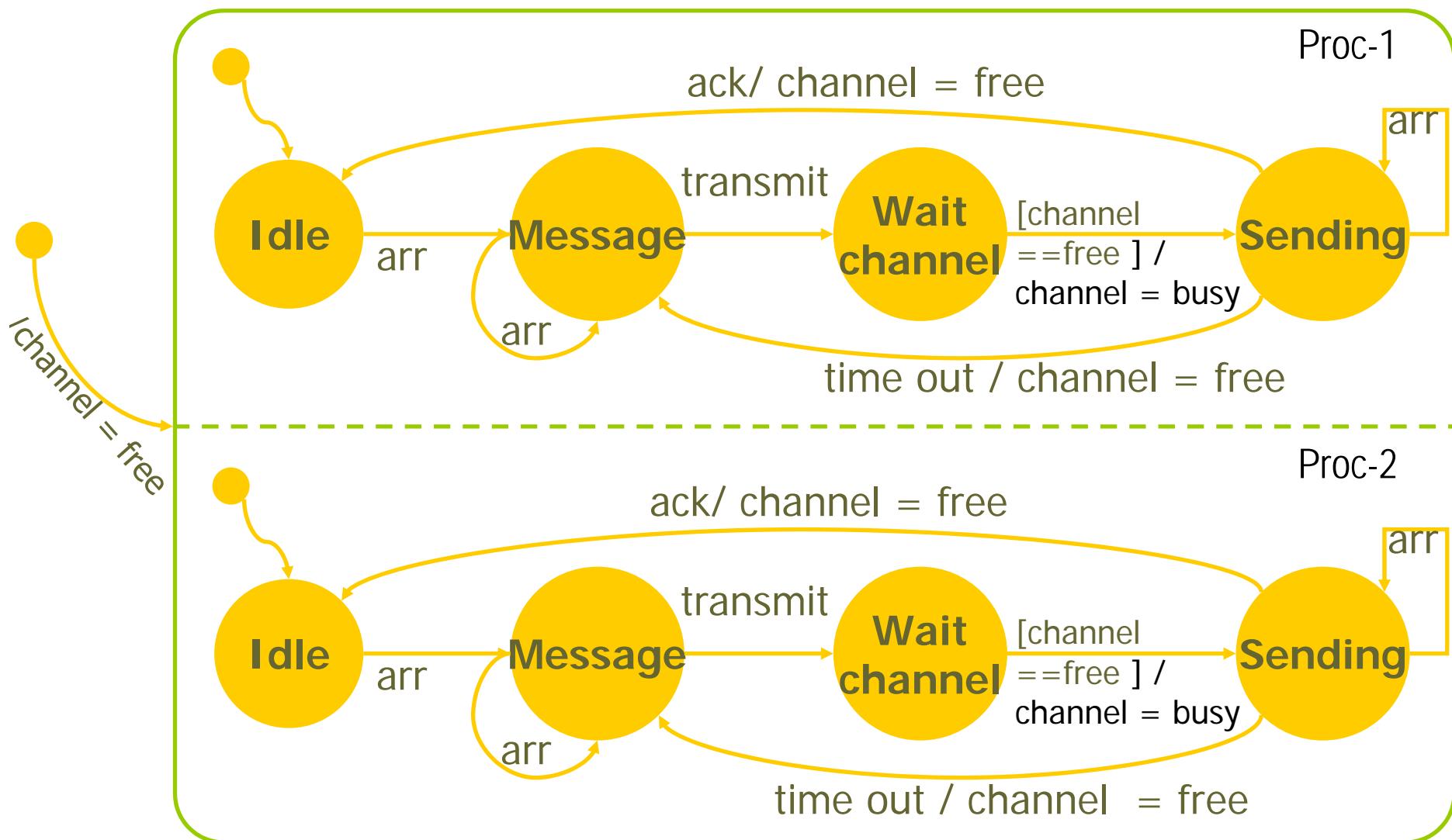
Máquinas de Estados

Componentes Ortogonales: Utilidad



Máquinas de Estados

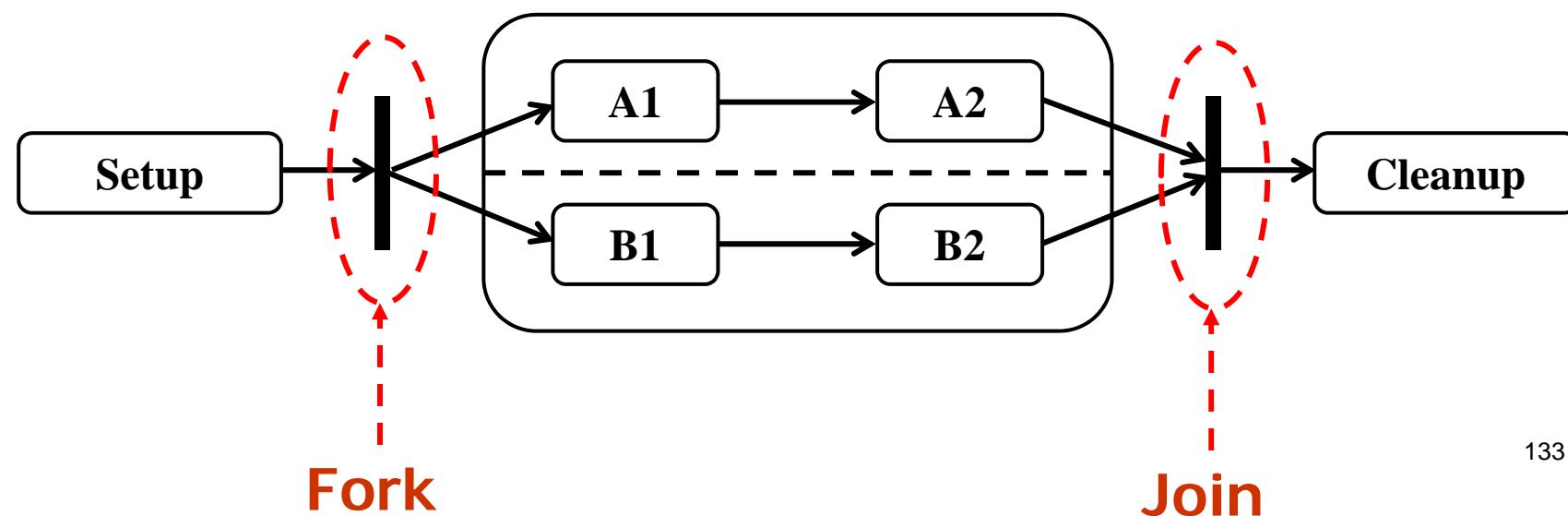
Componentes Ortogonales: Utilidad



Máquinas de Estados

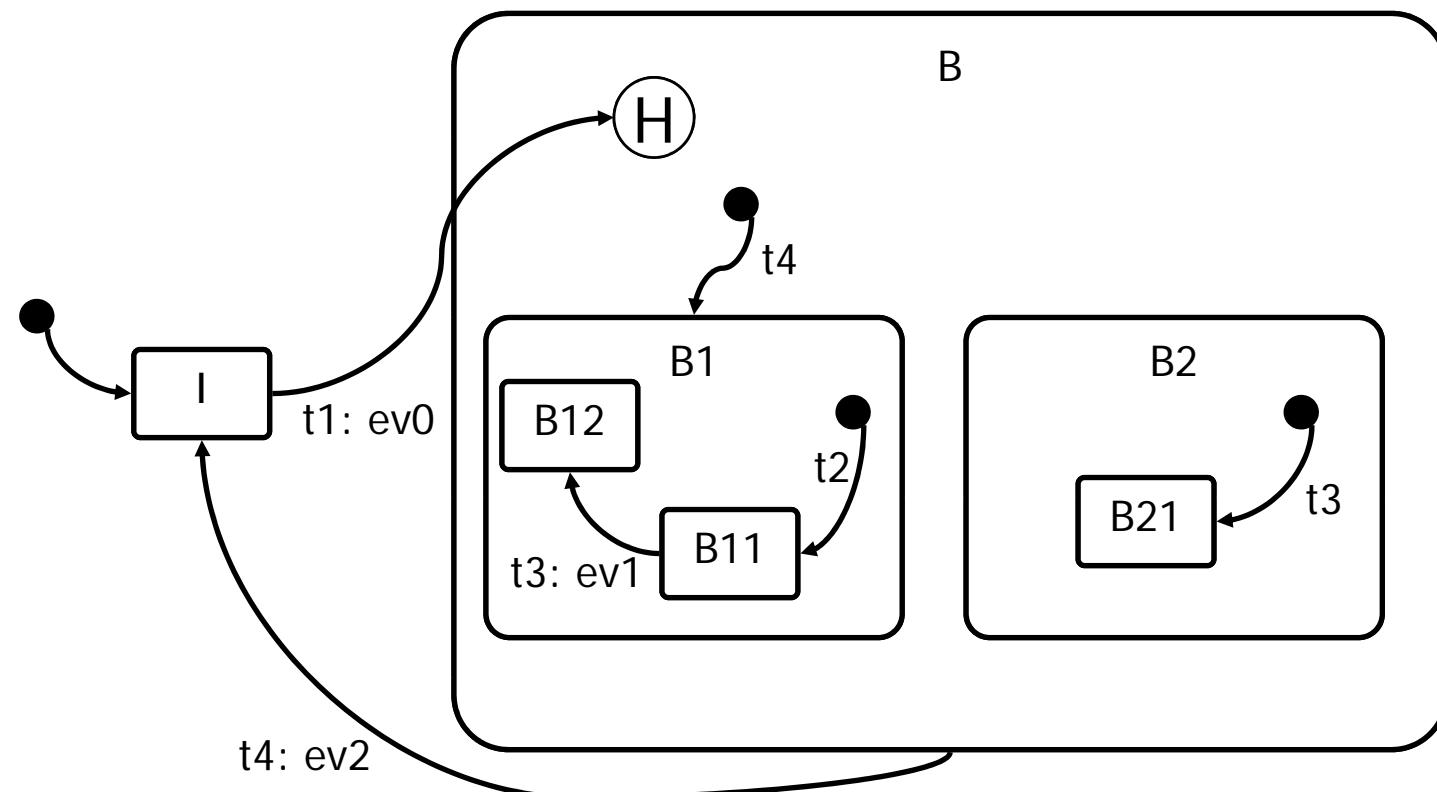
Pseudo - estados:

- Fork / Join.
- Initial.
- Deep History / Shallow History. H^* H
- Junction.
- Choice.
- Entry / Exit point.
- Terminate. \times



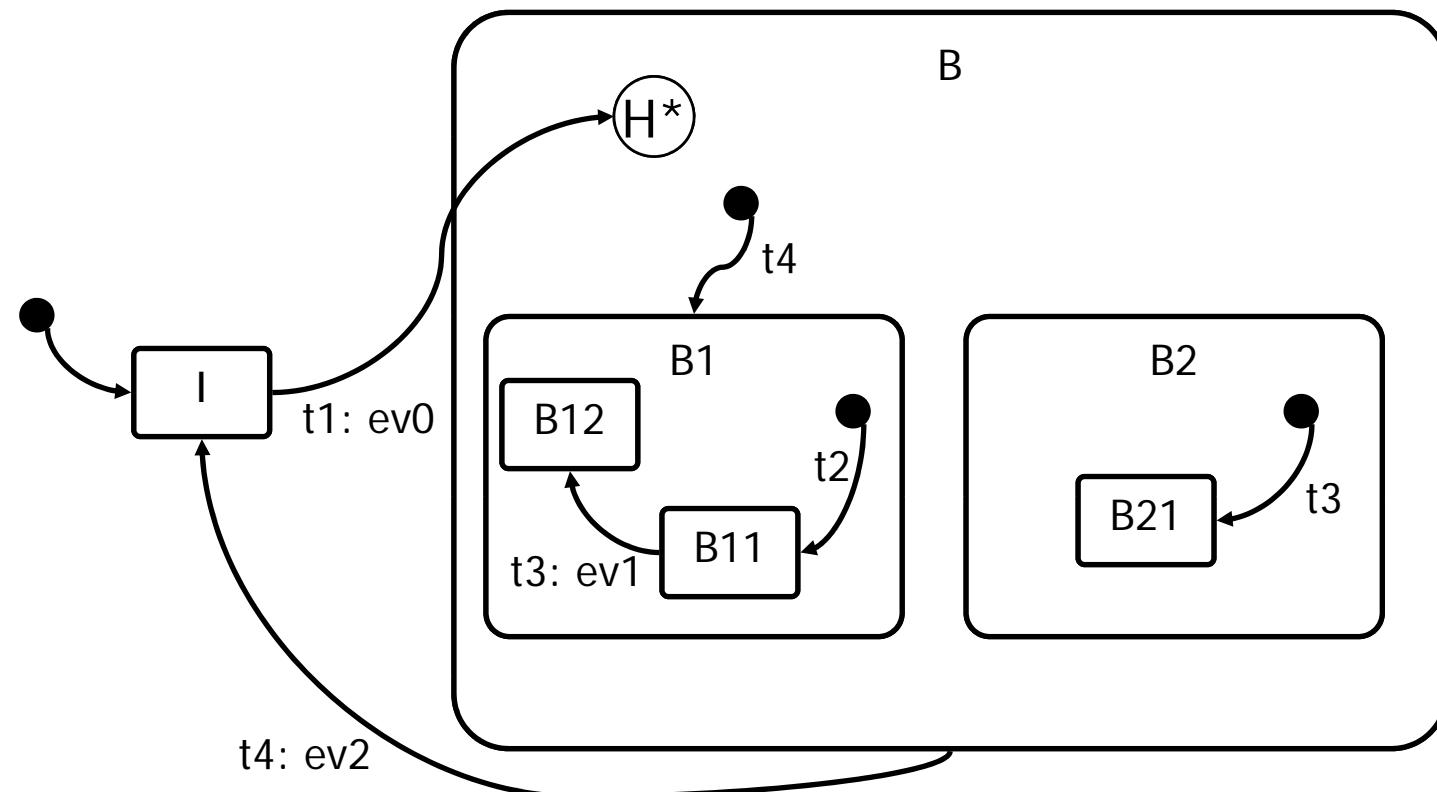
Máquinas de Estados

Estado Histórico



Máquinas de Estados

Estado Histórico (ii)



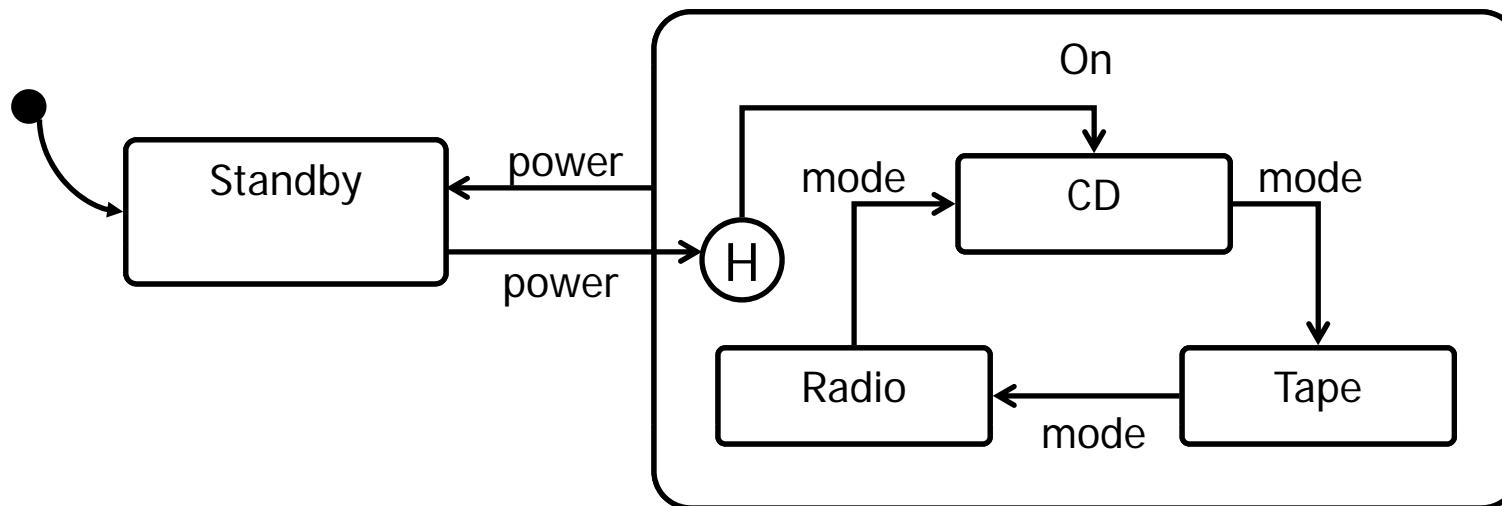
Máquinas de Estados

Estado Histórico. Ejercicio.

- Modelar el comportamiento de una cadena de música. Esta puede estar encendida (ON) o apagada (Standby). La cadena tiene reproductor de CD, Radio y Cinta. Se cambia de uno a otro con el botón “mode”. Cuando se enciende la cadena se recuerda el último estado en el que estuvo.

Máquinas de Estados

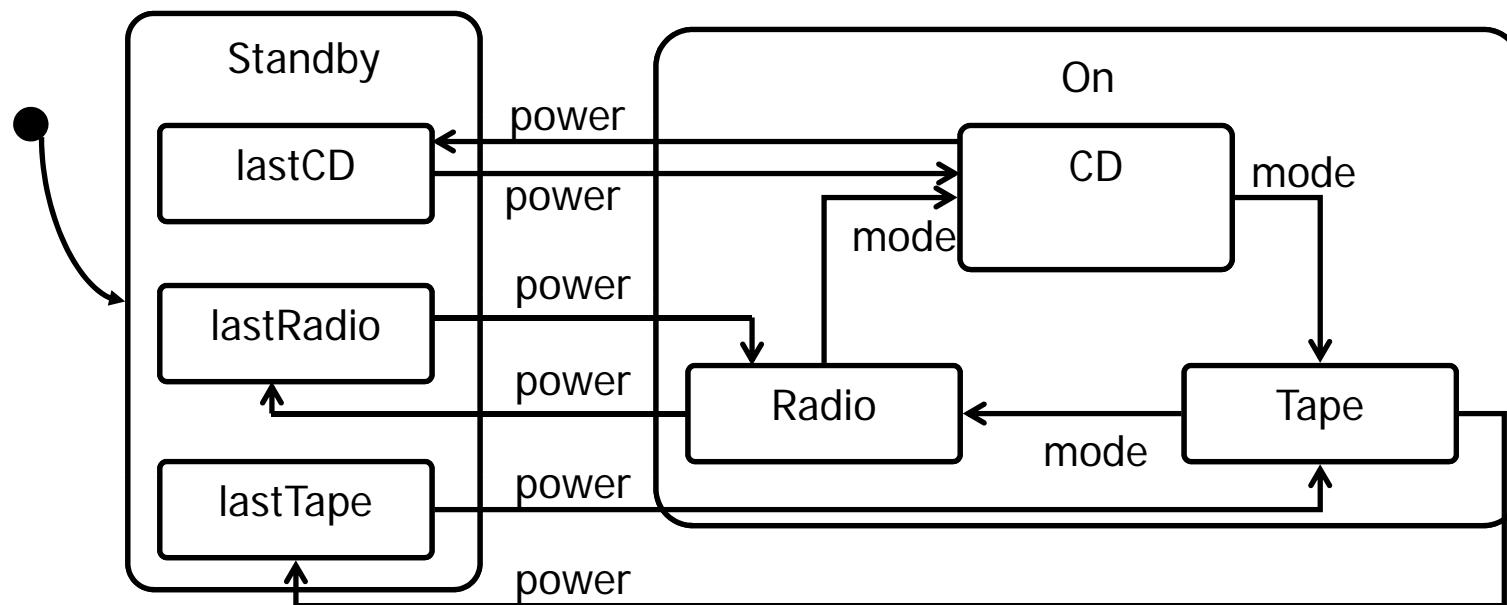
Estado Histórico. Ejercicio. Solución



Modelar el mismo sistema sin usar estado histórico.

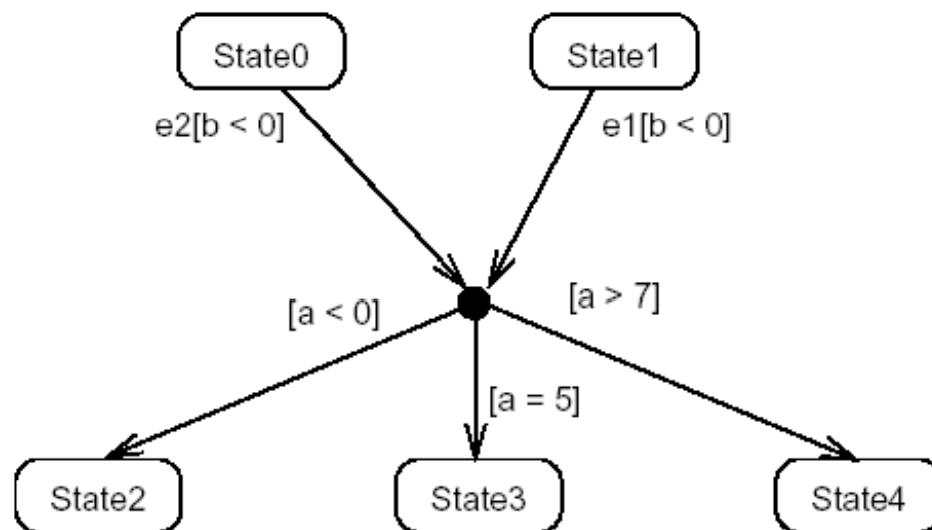
Máquinas de Estados

Estado Histórico. Ejercicio. Solución (ii)

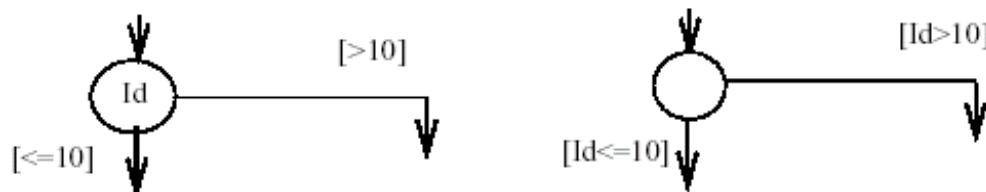


Máquinas de Estados

Pseudo - estados: Junction.

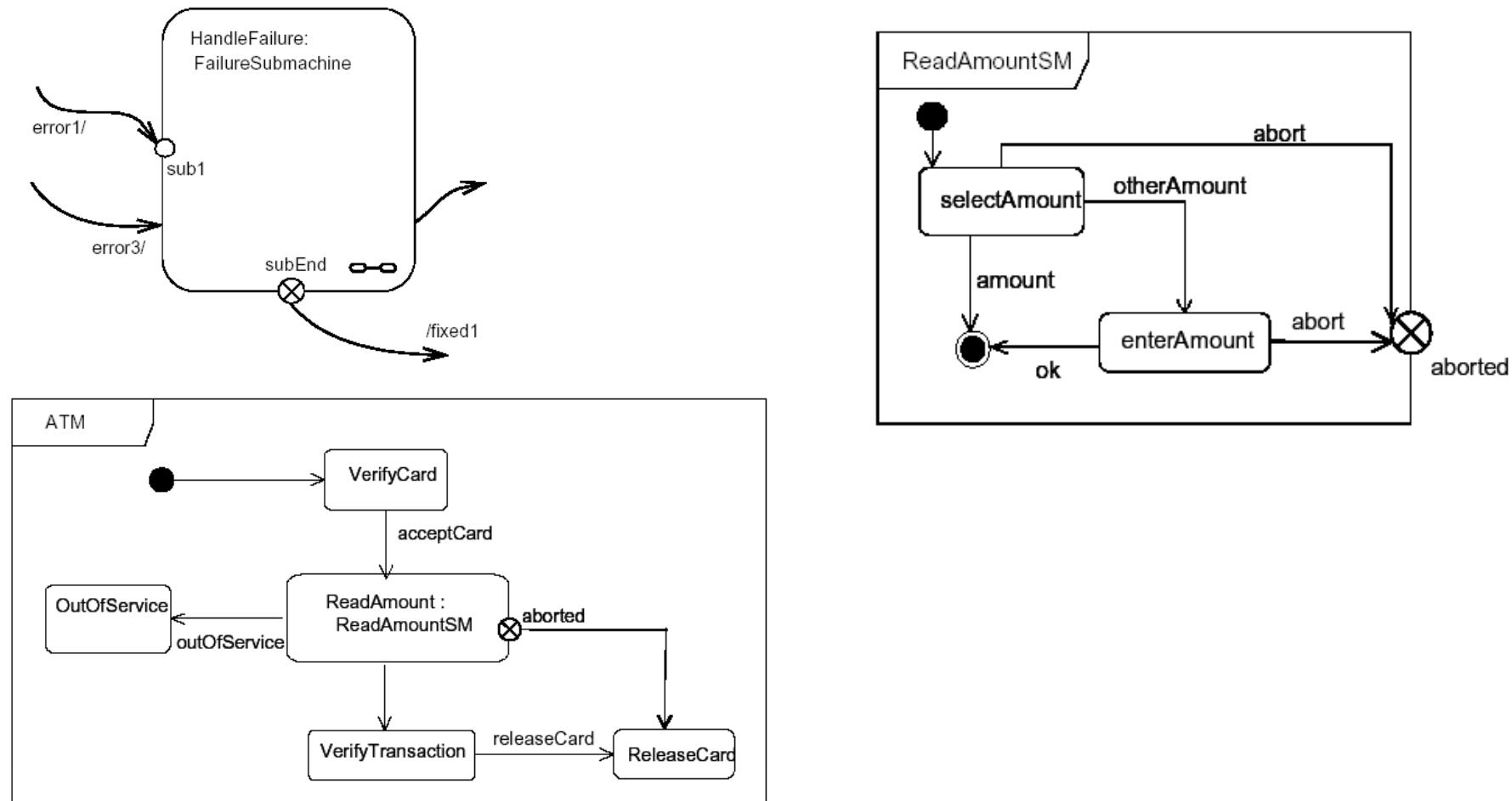


Pseudo - estados: Choice.



Máquinas de Estados

Puntos de Entrada/Salida, Estados Sub-Máquina



Ejemplo. Reproductor CDs.

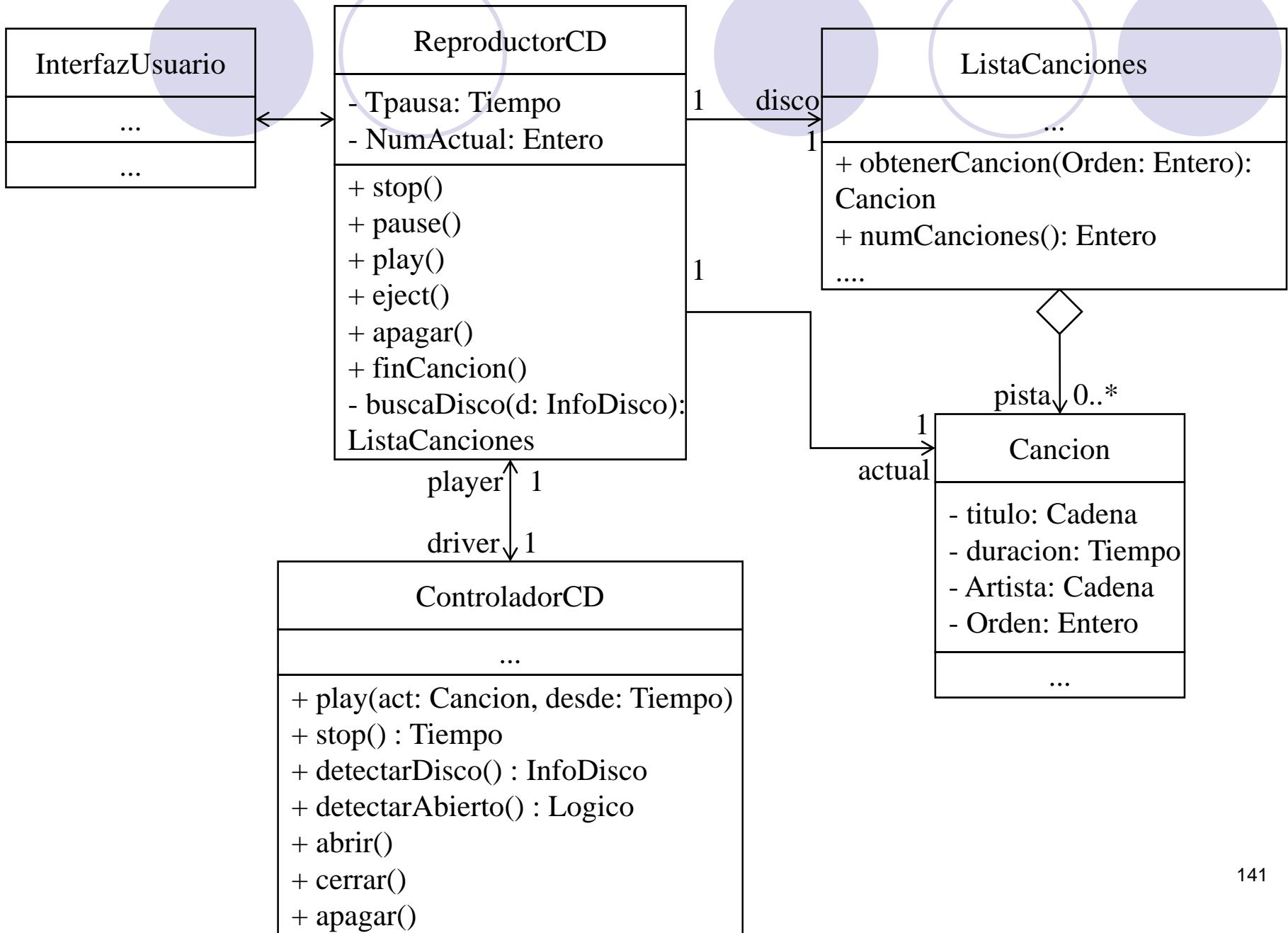
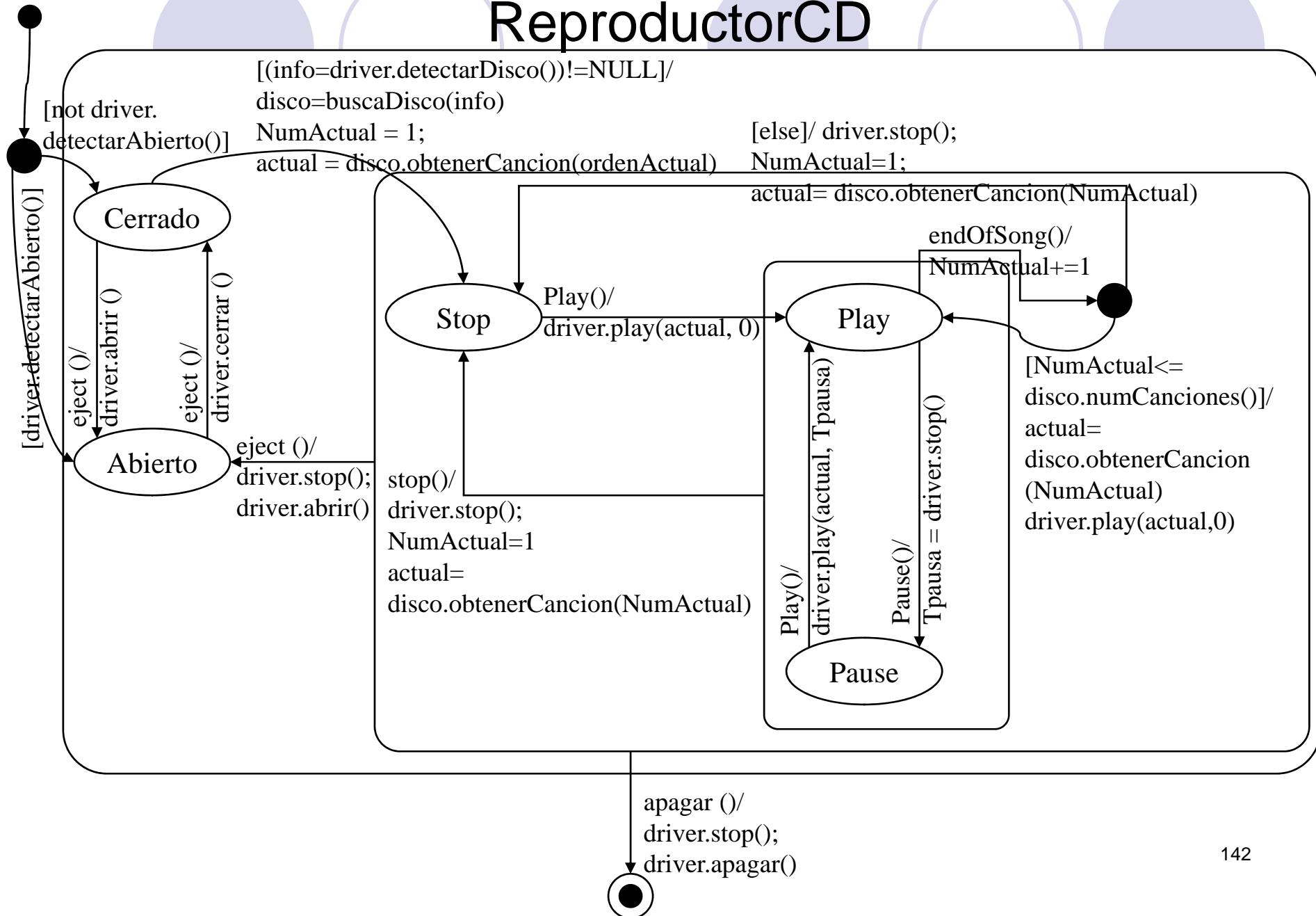


Diagrama de estados para la clase ReproductorCD

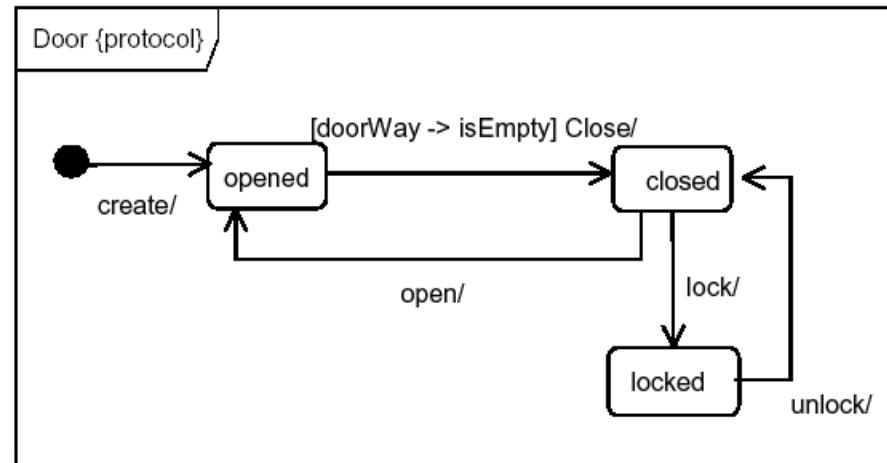


Máquinas de Estados de Protocolo

- Asociadas a un clasificador, interface o puerto.
- Especifican qué operaciones del clasificador se pueden llamar en qué condiciones.
- Las operaciones que no generan transición no se representan.
- ***Máquinas de estado declarativas:*** Especifican las transiciones legales (no su condición) para cada operación. “Contrato” para el usuario del clasificador.
- ***Máquinas de estado Ejecutables:*** Especifican todos los eventos que un clasificador puede recibir y tratar, con las transiciones implicadas.

Máquinas de Estados de Protocolo

- Máquinas de estados de protocolo. Ejemplo (declarativa).



- Los estados de máquinas de protocolo no tienen asociadas acciones exit/entry/do, pero pueden tener invariantes.
- Las transiciones no tienen acciones, pero sí pueden tener pre- y post- condiciones.

Cleanup
[inv]

[pre-cond] Evt/ [post-cond]

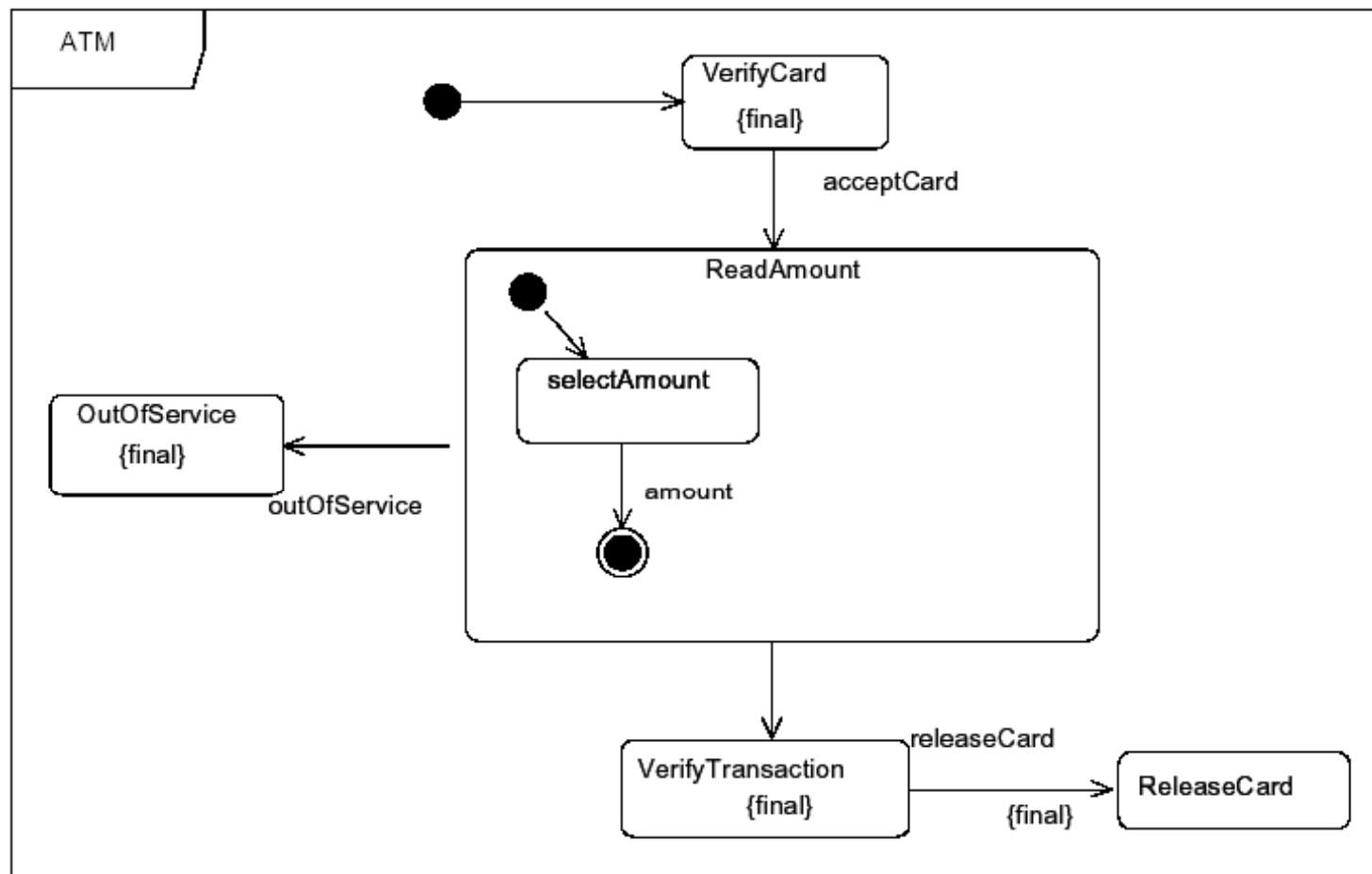
Máquinas de Estados

Generalización

- Una máquina de estados es generalizable.
- Se pueden añadir regiones, estados y transiciones.
- Se puede cambiar el destino y estado de una transición siempre que la fuente y evento se mantenga.
- En caso de herencia múltiple de máquinas de estado (por herencia de los clasificadores asociados), se crea una región ortogonal por cada máquina heredada, mas una por la máquina de estados del clasificador específico.
- Se anota con <> junto al nombre de la máquina.
- Los estados heredados se muestran con líneas punteadas o en gris.

Máquinas de Estados

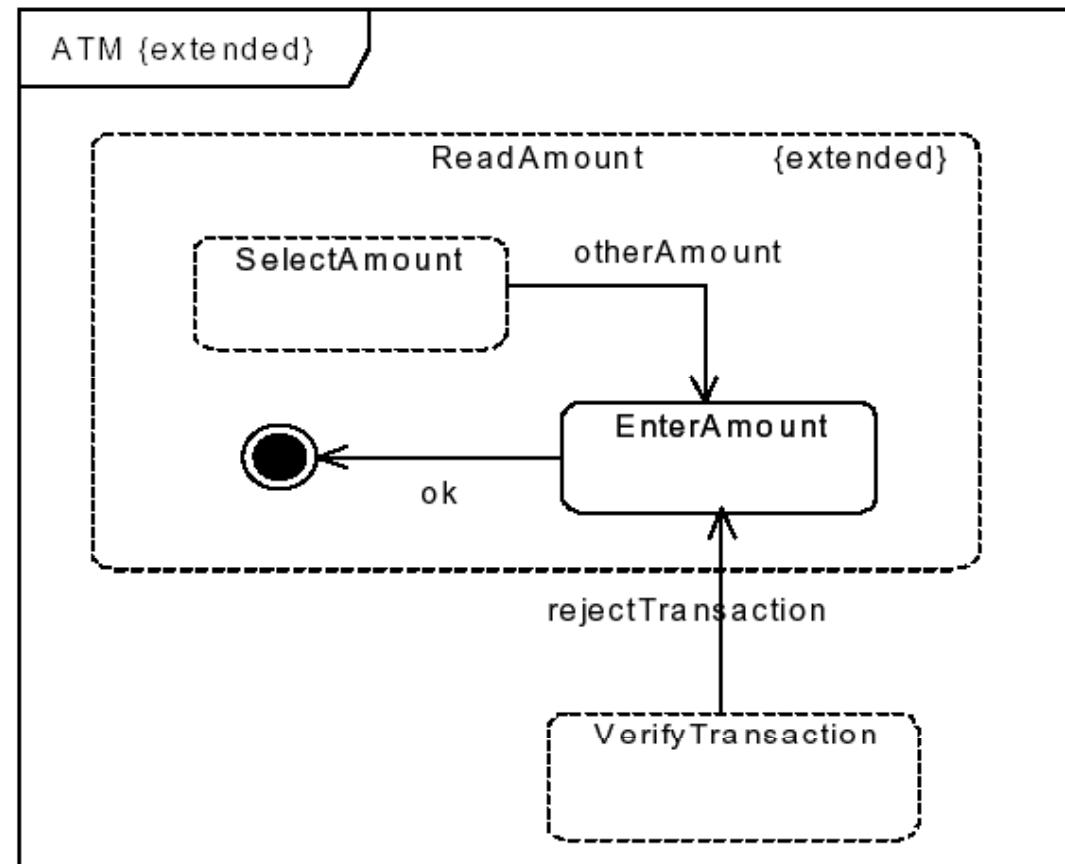
Generalización: Ejemplo, cajero automático.



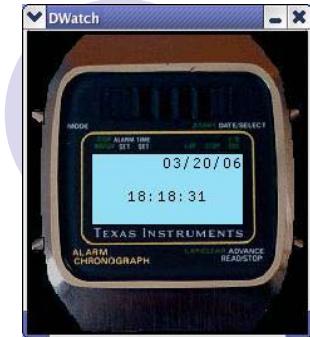
Máquinas de Estados

Generalización: Ejemplo, cajero automático.

Extensión: posibilidad de teclear el importe a retirar, y de que este se pueda rechazar.



Ejercicio

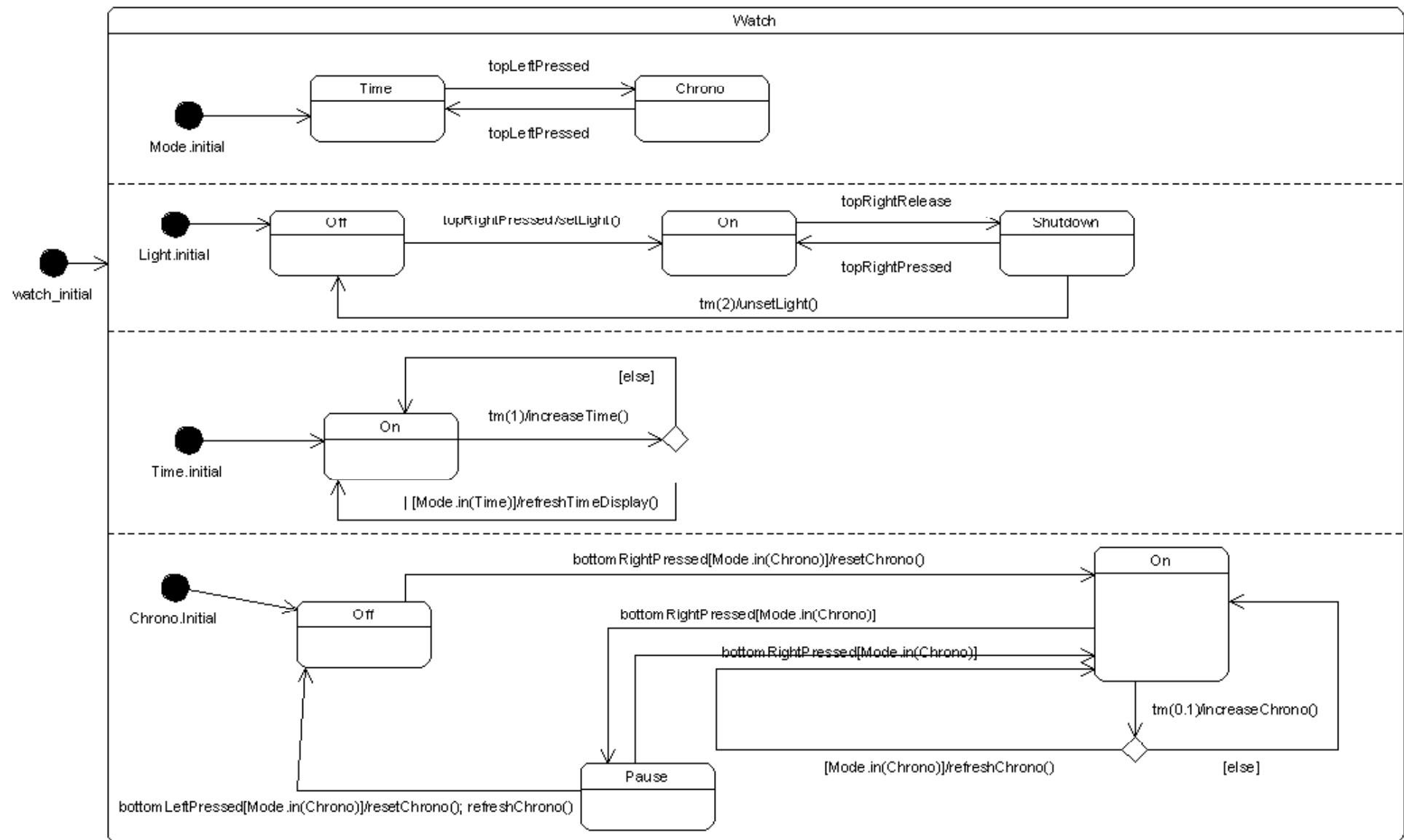


- Modelar el comportamiento reactivo de un reloj de pulsera.
- El valor del tiempo se debe actualizar cada segundo, incluso cuando no se muestra (p.ej. crono encendido).
- El botón de la parte superior derecha enciende la luz, que se mantiene encendida tanto como el botón está apretado, una vez que se suelta, la luz está encendida durante 2 segundos más y se apaga.
- El botón superior izquierdo alterna entre el modo de crono y de reloj. El sistema empieza en el modo reloj, en el que se muestra la hora en formato HH:MM:SS.
- En el modo crono, el tiempo transcurrido se muestra en formato MM:SS:CC (CC son centésimas de segundo). Inicialmente el crono empieza en 00:00:00. El botón inferior derecho se usa para activar el crono. Éste se actualiza en incrementos de 1/100 segundos. Presionando el botón inferior derecho pausa o continua el crono (si el reloj está en modo crono). Pulsando el botón inferior izquierdo resetea el crono a 00:00:00 si el reloj está en modo crono y el crono ha sido pausado antes. El crono continua corriendo (si está corriendo) o mantiene su valor (si está en pausa) incluso cuando el reloj está en un modo de display distinto (por ejemplo, cuando se muestra la hora).

Ejercicio

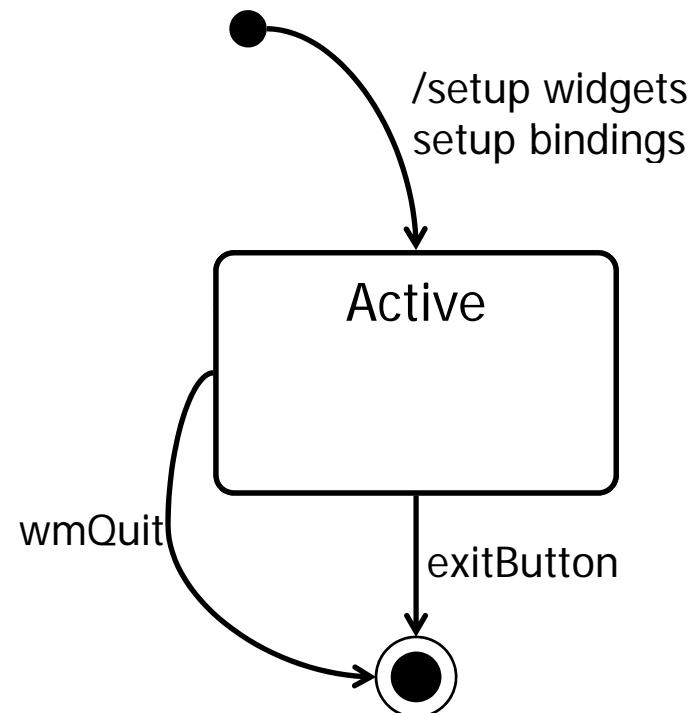
- Interface provisto por el controlador:
 - ***getTime()*** : Devuelve la hora actual.
 - ***refreshTimeDisplay()*** : Repinta la hora en el visor con la hora interna actual. El visor no necesita limpiarse antes de llamar a esta función. Por ejemplo, si se está visualizando el crono, se borrará antes de pintar la hora.
 - ***refreshChronoDisplay()*** : ver ***refreshTimeDisplay()***.
 - ***resetChrono()*** : Resetea el crono interno a 00:00:00.
 - ***increaseTime()*** : Incrementa la hora en un segundo. Los minutos y horas se modificarán adecuadamente, (por ejemplo, si se llama a ***increaseTime ()*** a las 11:59:59, la nueva hora será 12:00:00).
 - ***increaseChrono ()*** : Incrementa el crono en 1/100 segundos.
 - ***setLight()*** : Enciende la luz del visor.
 - ***unsetLight()*** : Apaga la luz del visor.
- Eventos de botones recibidos:
 - ***topRightPressed.***
 - ***topRightReleased.***
 - ***topLeftPressed.***
 - ***topLeftReleased.***
 - ***bottomRightPressed.***
 - ***bottomRightReleased.***
 - ***bottomLeftPressed.***
 - ***bottomRightReleased.***

Possible Solución.



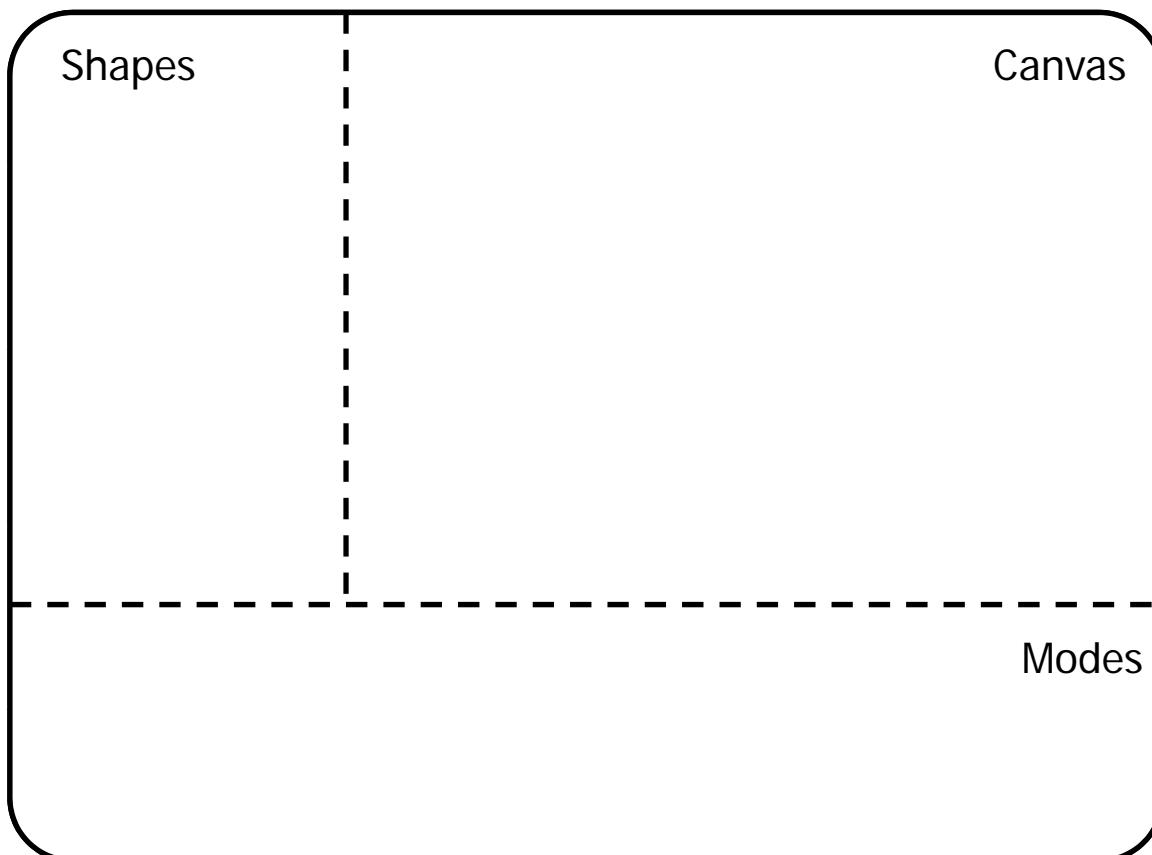
Máquinas de Estados

Ejemplo. Herramienta de Dibujo (i)



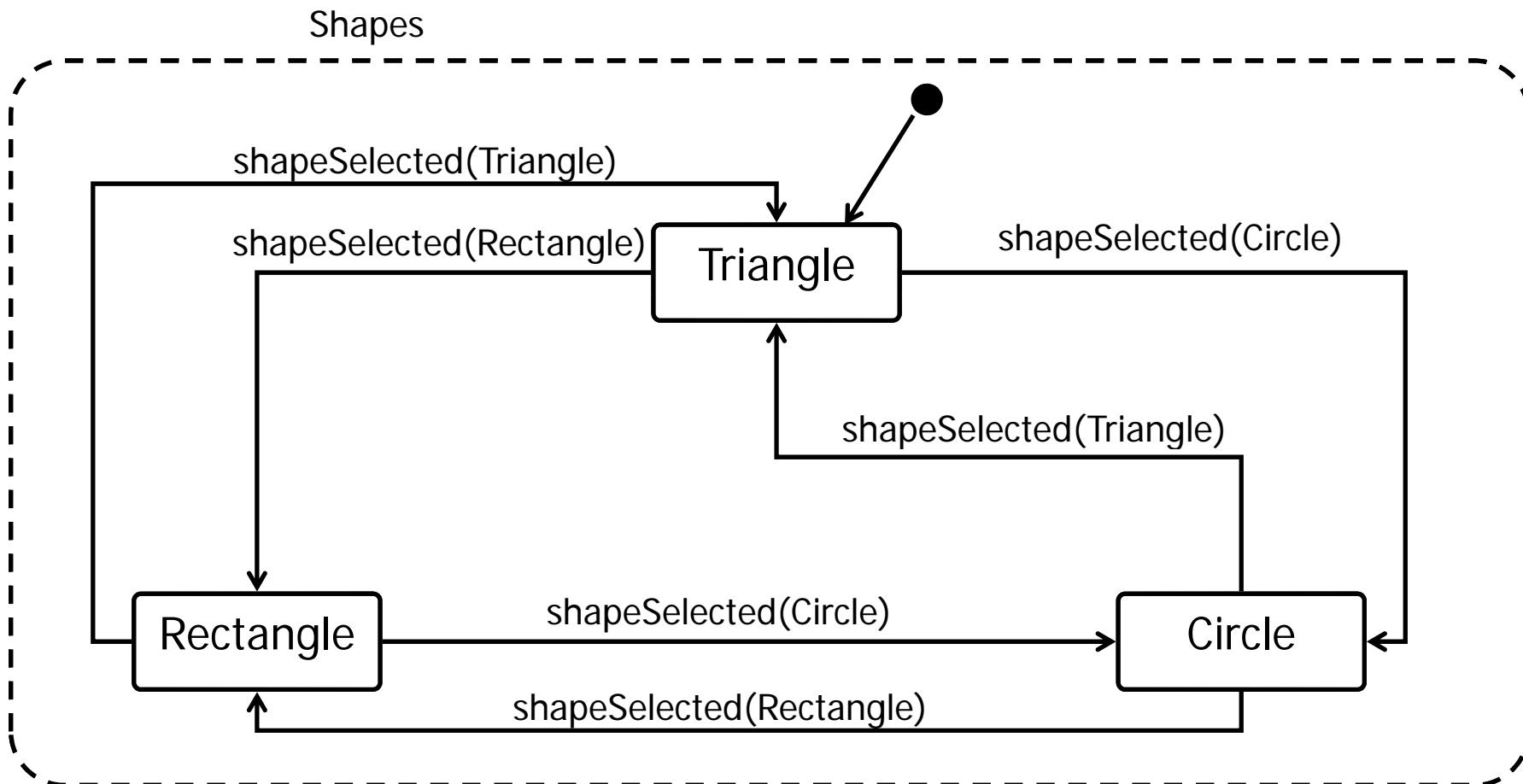
Máquinas de Estados

Ejemplo. Herramienta de Dibujo (ii)



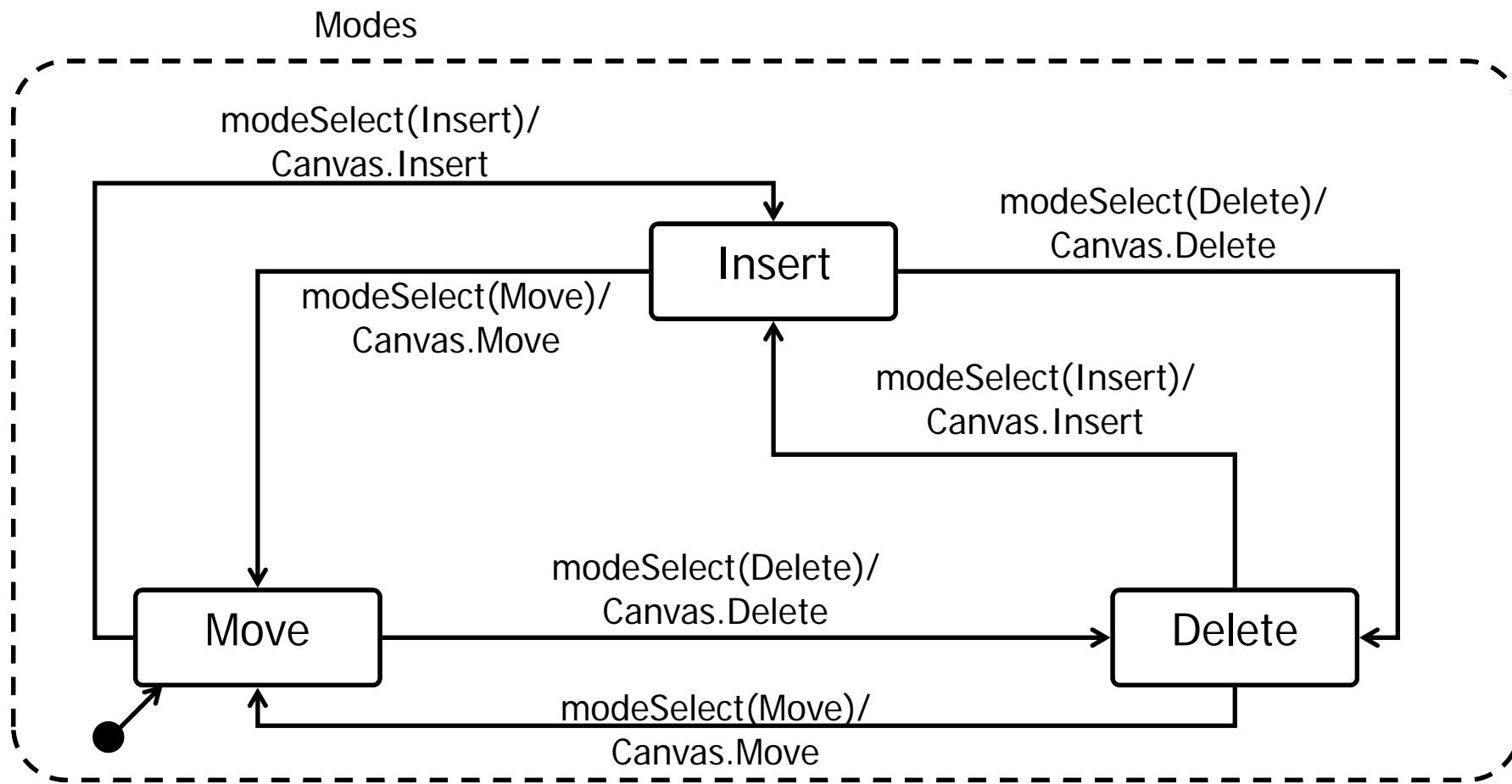
Máquinas de Estados

Ejemplo. Herramienta de Dibujo (iii)



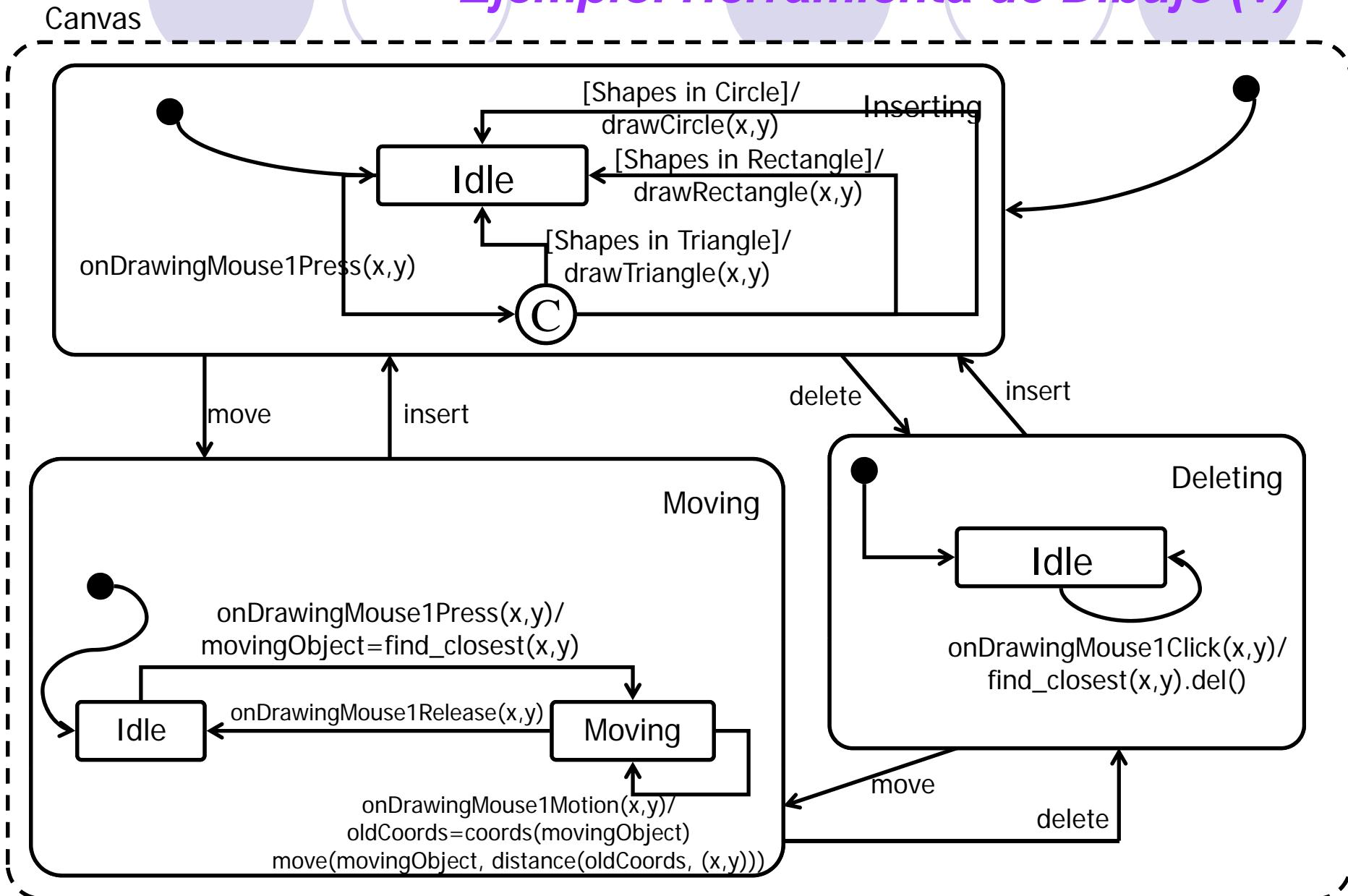
Máquinas de Estados

Ejemplo. Herramienta de Dibujo (iv)



Máquinas de Estados

Ejemplo. Herramienta de Dibujo (v)

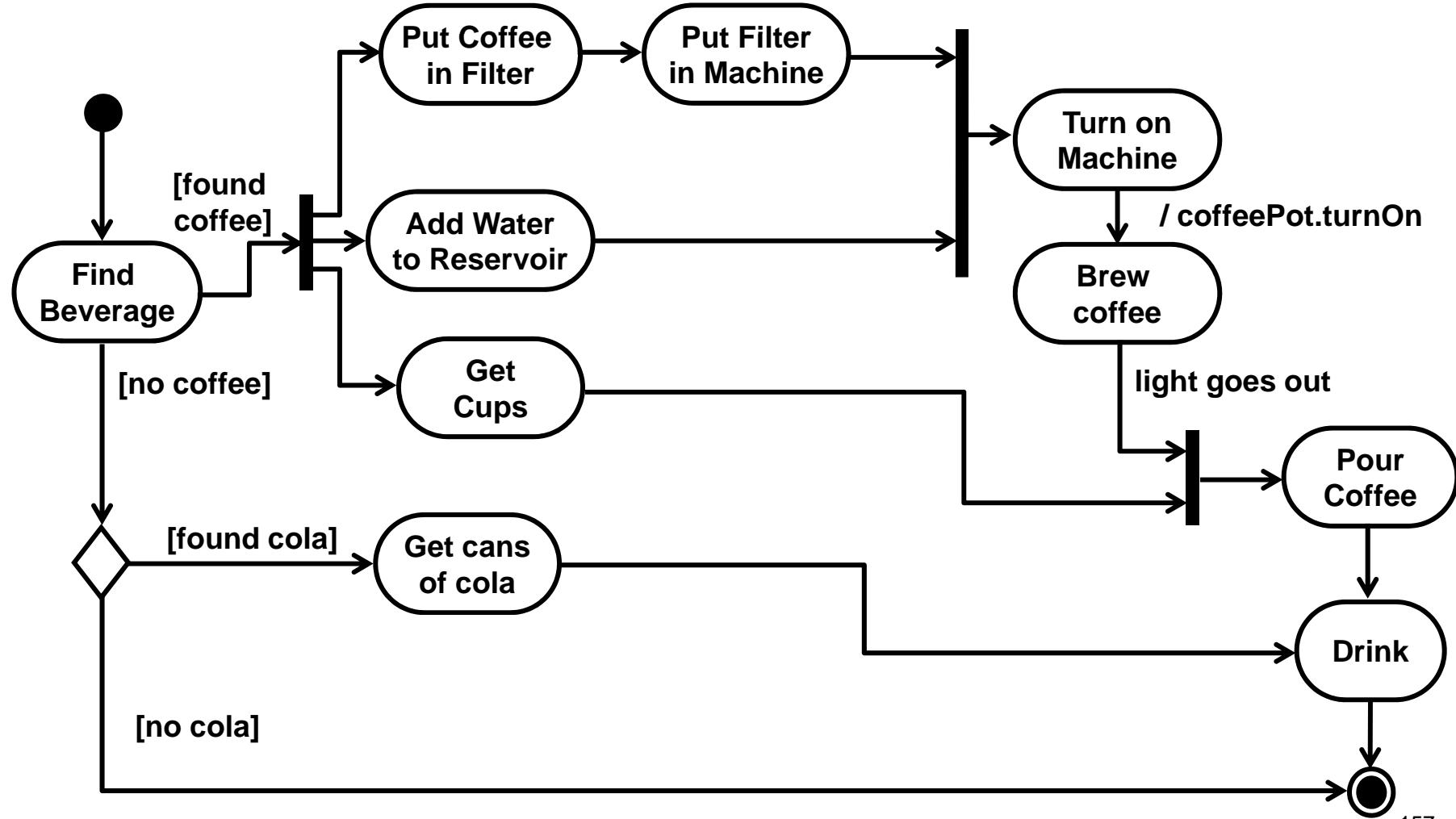


Diagramas de Actividad

- Refinamiento de los diagramas de estados.
 - Los estados representan la ejecución de acciones o subactividades
 - Las transiciones son disparadas cuando se completan estas acciones o subactividades
 - Semántica basada en tokens.
- Flujos dirigidos por procesamiento interno (en los diagramas de estados normales son dirigidos por eventos externos).
- Semántica basada en Redes de Petri. No obstante no se da una transformación a Redes de Petri.

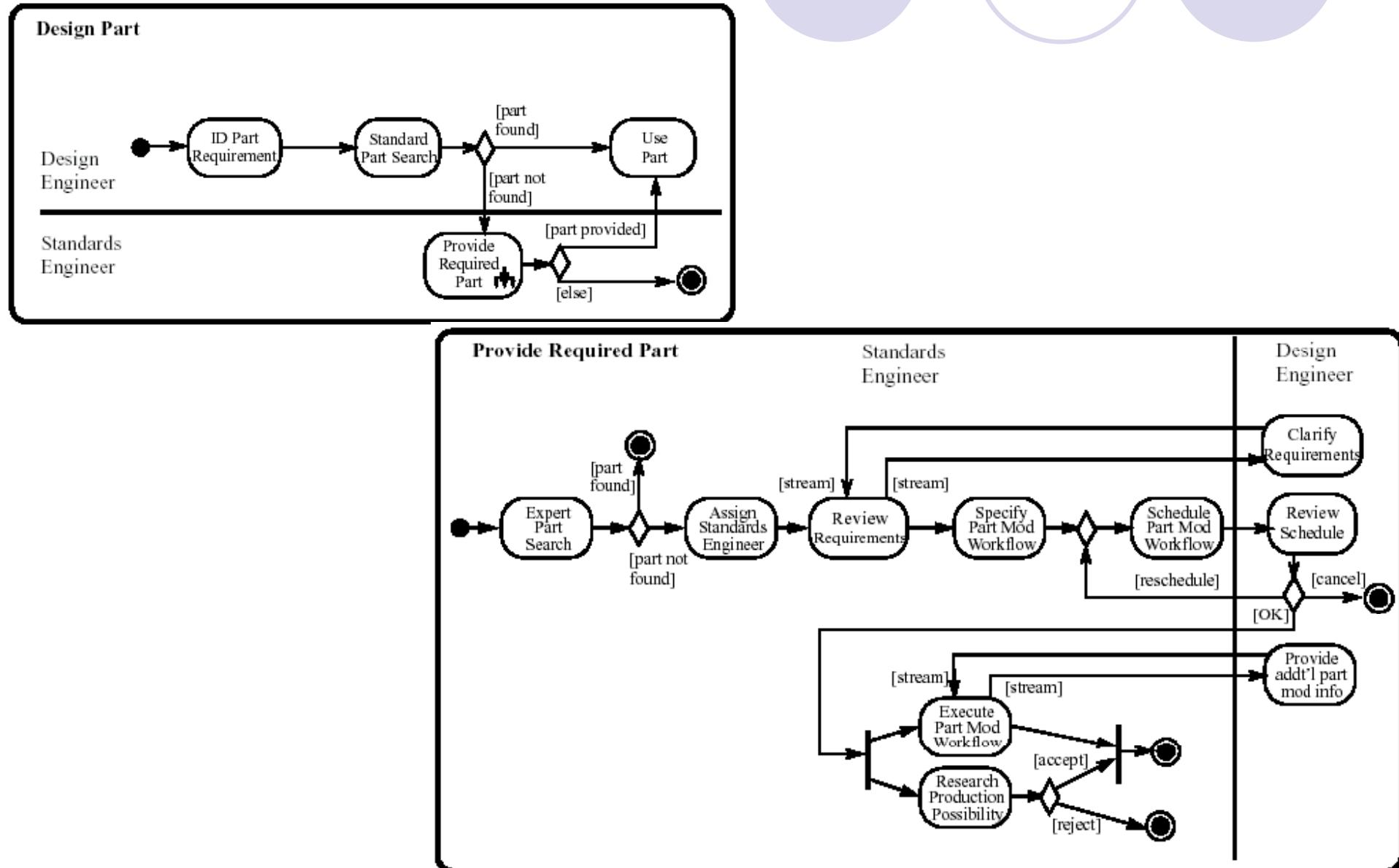
Diagramas de Actividad

Ejemplo



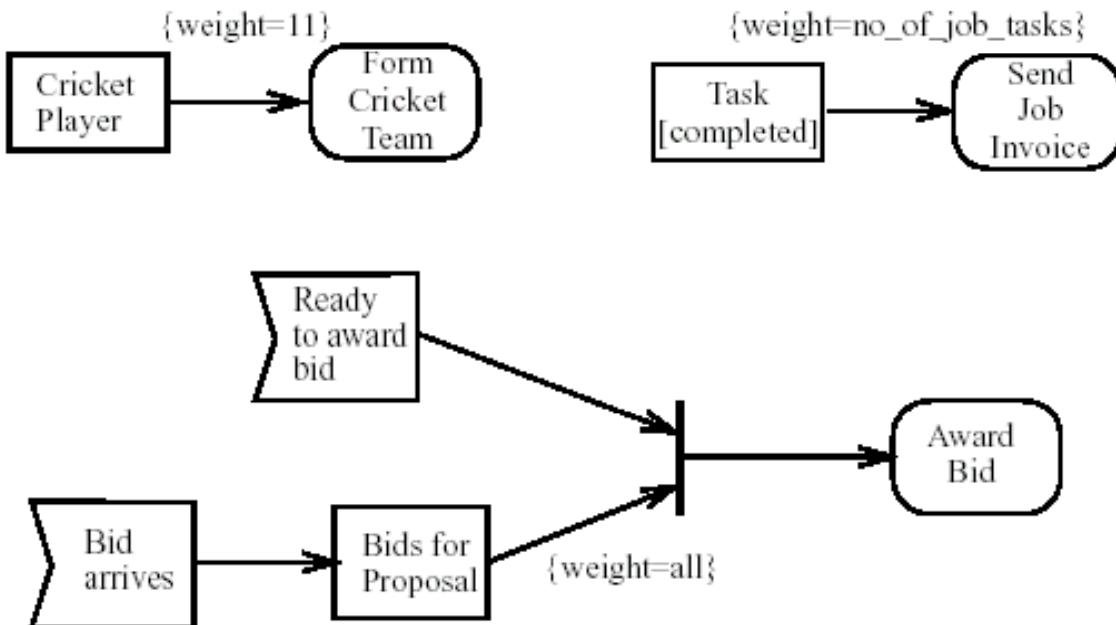
Diagramas de Actividad

Swimlanes



Diagramas de Actividad

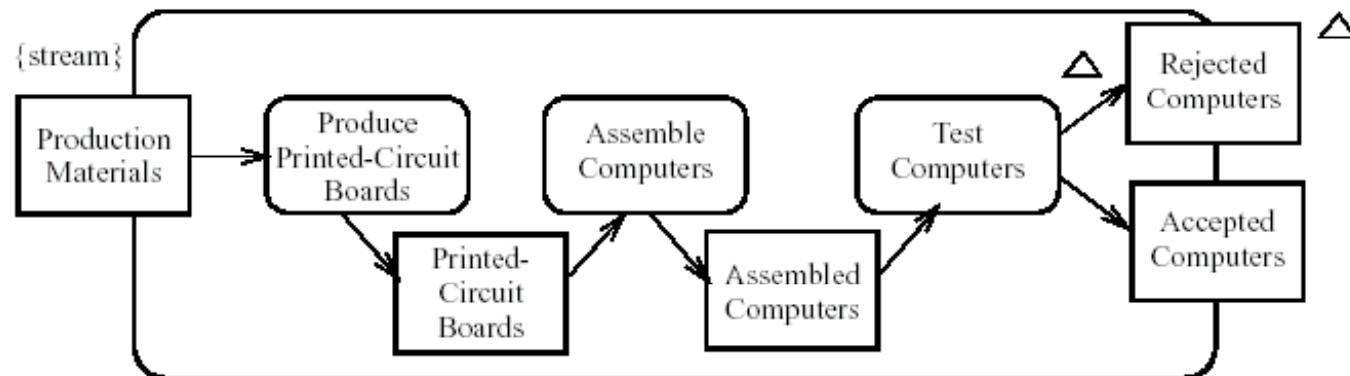
Pesos en los enlaces



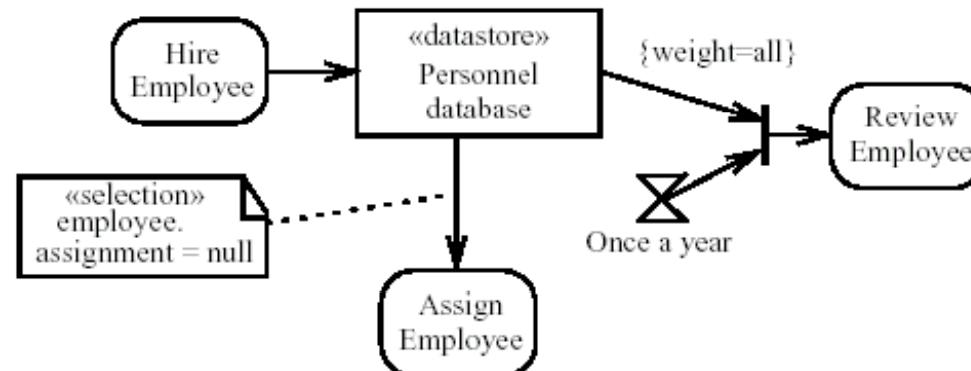
Diagramas de Actividad

Parámetros y Eventos Temporales

Parámetros/Pins/Excepciones

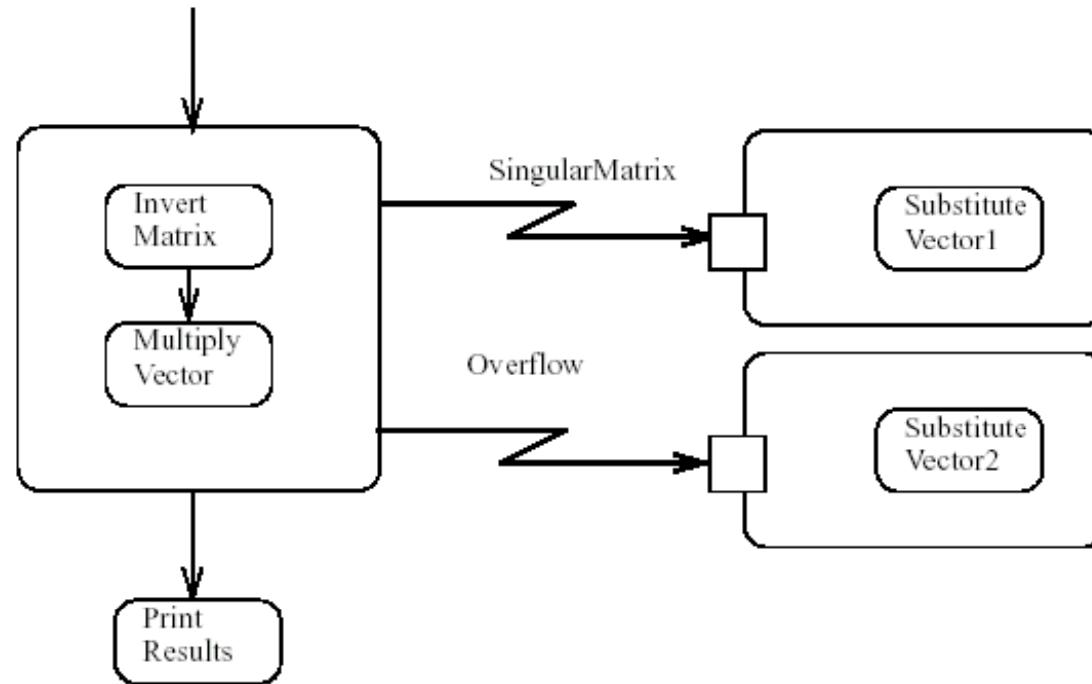


Eventos Temporales



Diagramas de Actividad

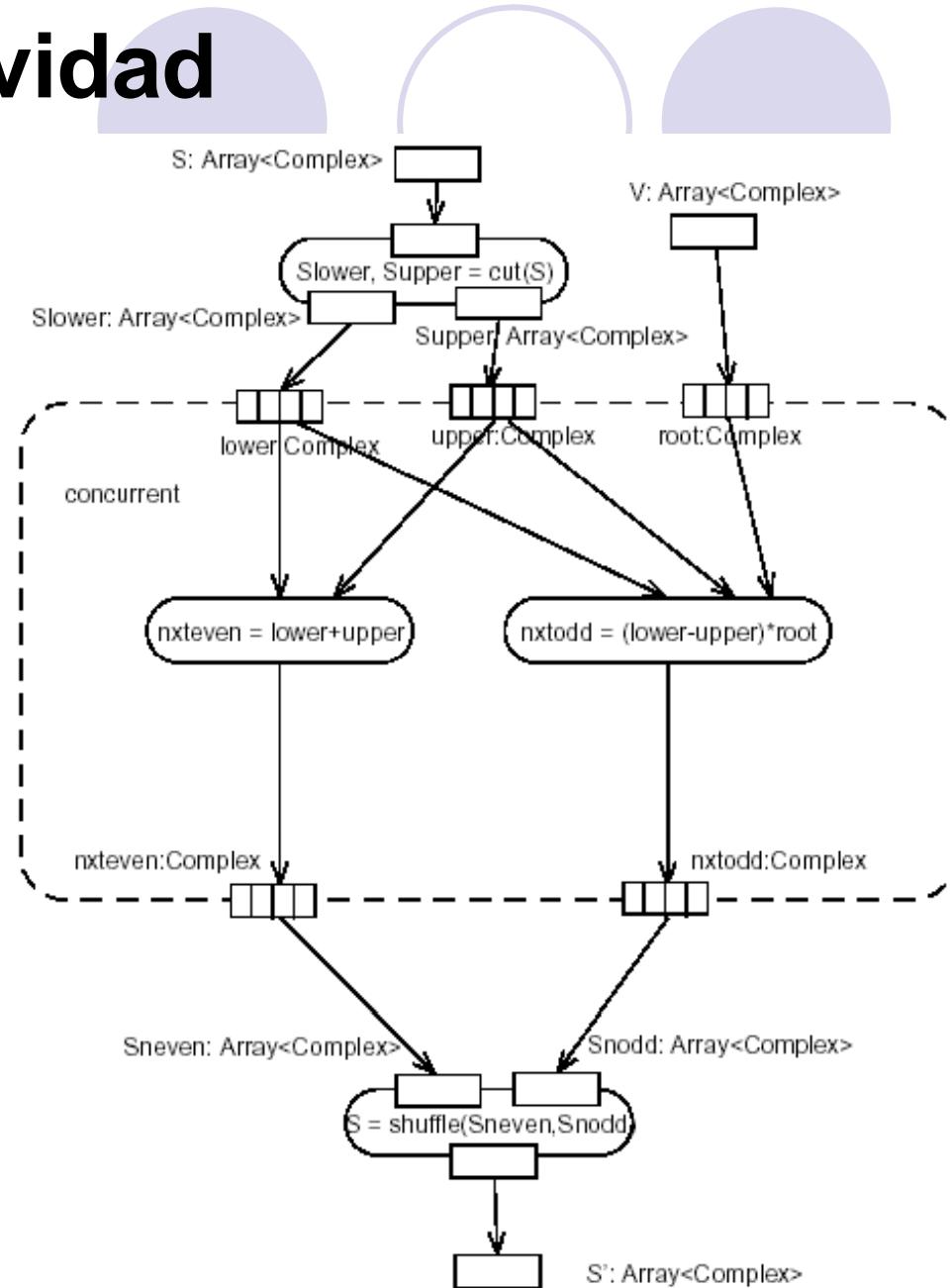
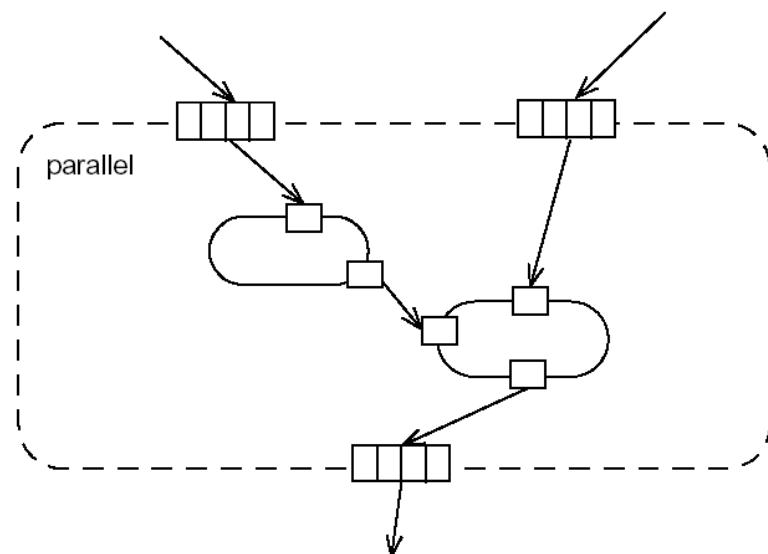
Excepciones/Pins



Diagramas de Actividad

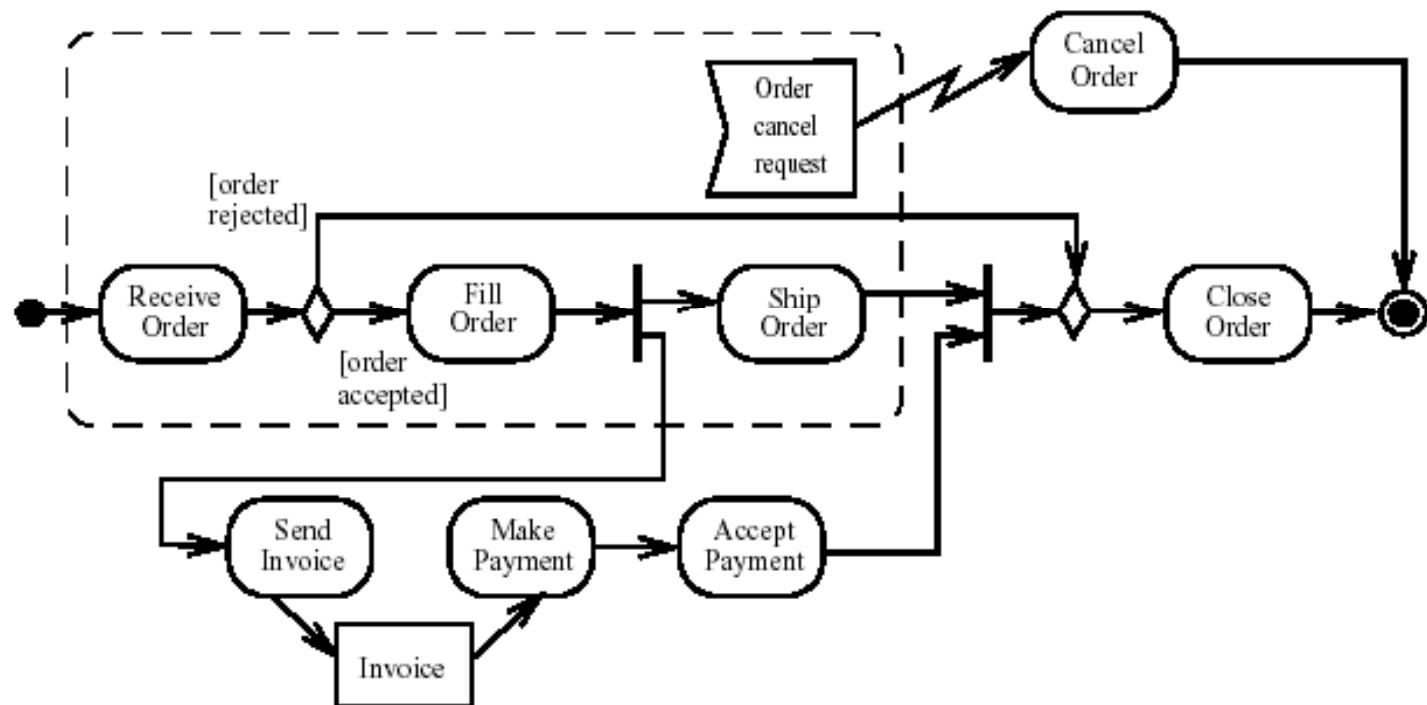
Regiones de Expansión

- Regiones de expansión, procesamiento paralelo (también *iterative* y *streaming*).



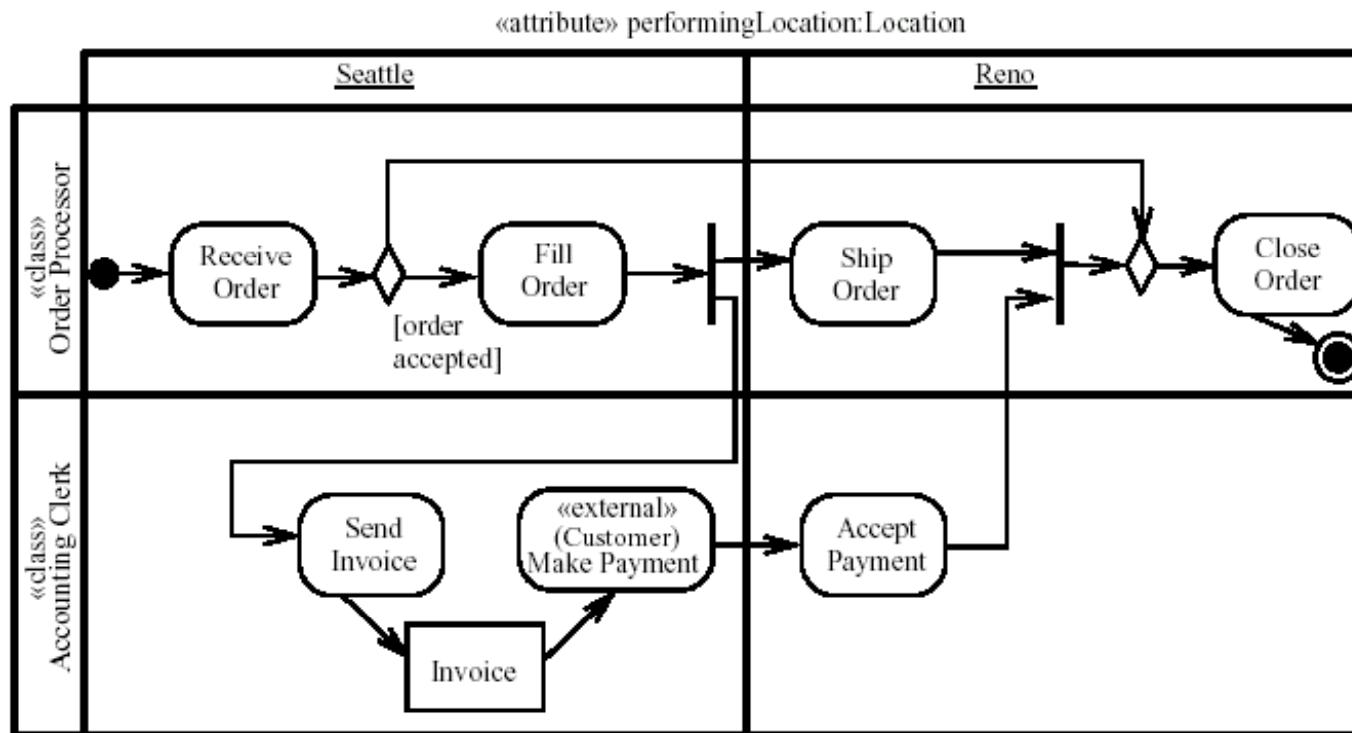
Diagramas de Actividad

Regiones Interrumpibles



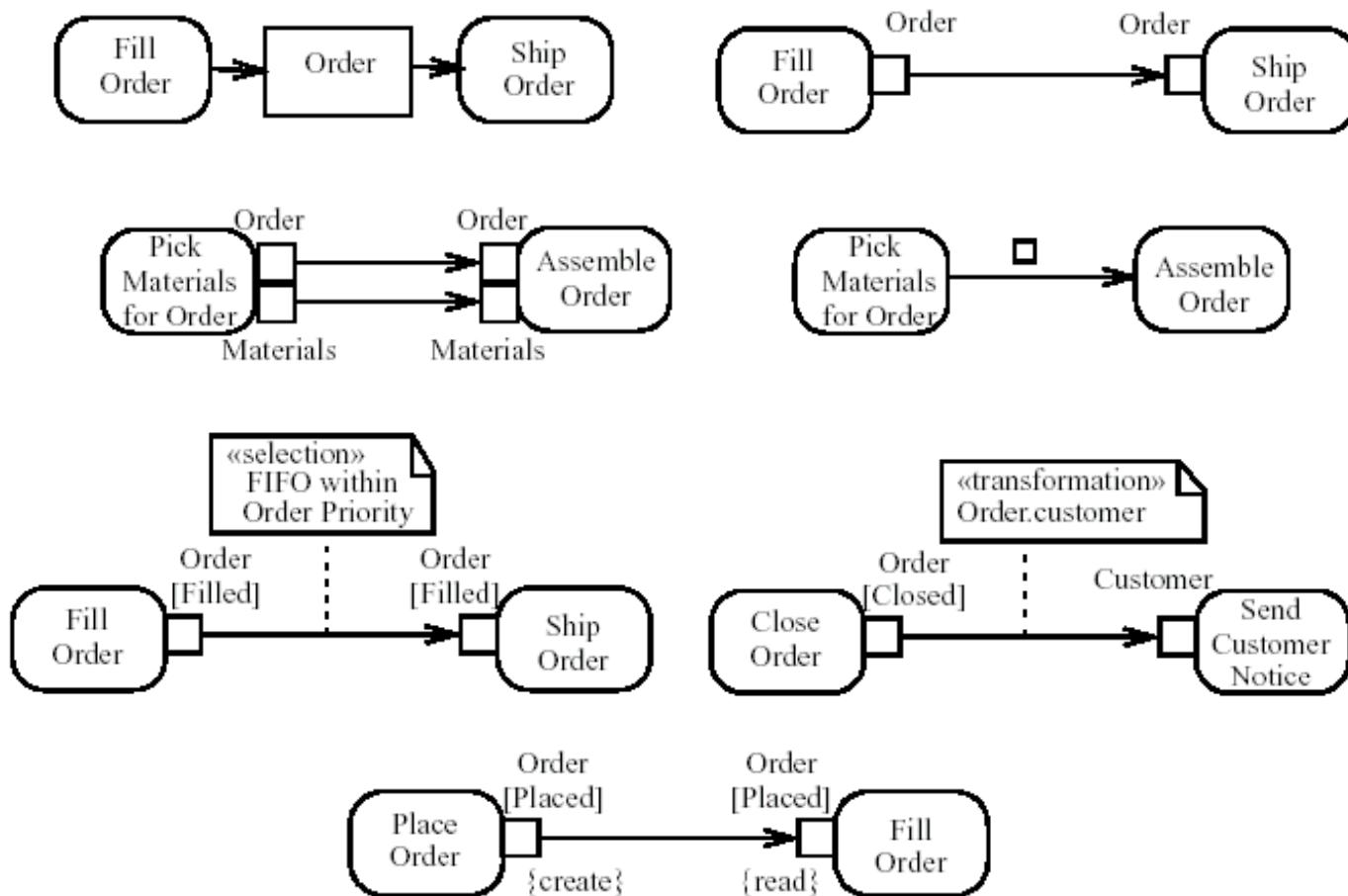
Diagramas de Actividad

Particiones



Diagramas de Actividad

Flujos de Objetos: Objectflows



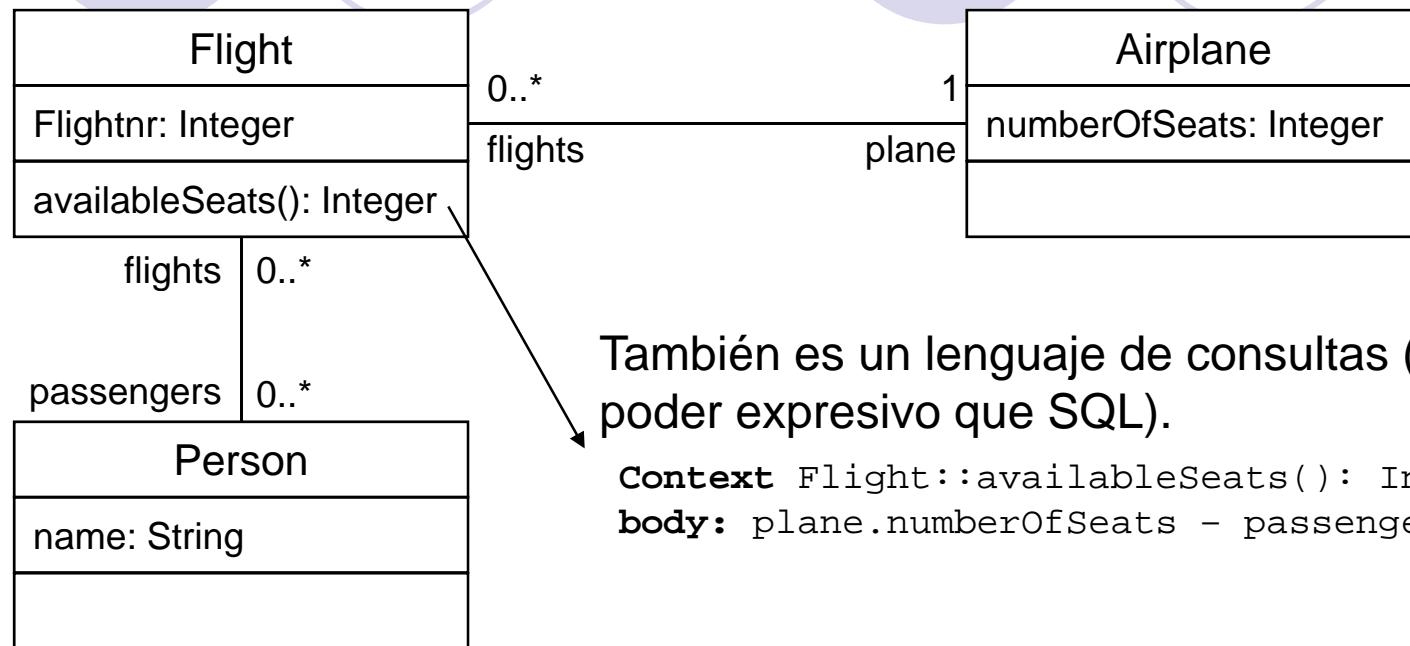
Indice

- Diagramas de Casos de Uso.
- Diagramas de Estructura.
- Diagramas de Comportamiento.
- **OCL.**
- Herramientas.
- Ejemplos.
- Bibliografía.

OCL: Object Constraint Language

- Lenguaje de restricciones para expresar condiciones adicionales que no podemos expresar con diagramas y cardinalidades.
- Combinar diagramas y especificaciones textuales.
- Lenguaje preciso, no ambiguo, declarativo, tipado, basado en matemáticas (lógica de predicados y teoría de conjuntos).
- Útil para obtener modelos precisos (no anotaciones en lenguaje natural).
- Se utiliza fundamentalmente junto a los diagramas de clases.

Ejemplos



También es un lenguaje de consultas (mismo poder expresivo que SQL).

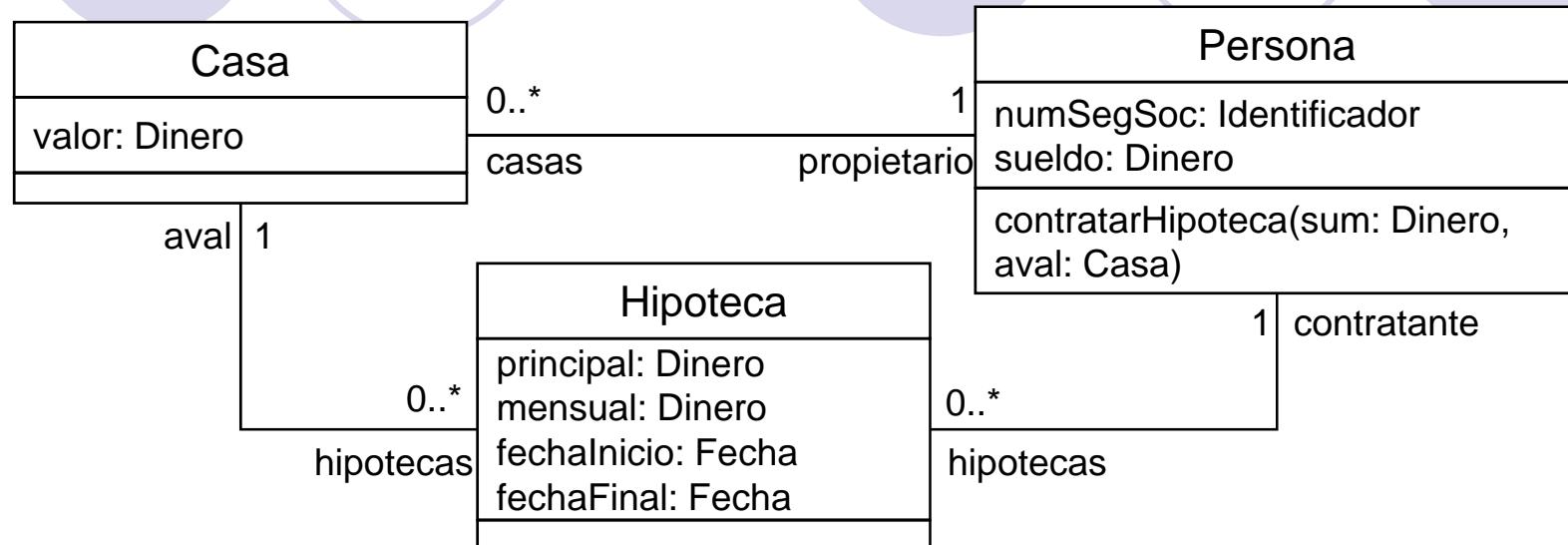
```
Context Flight::availableSeats(): Integer  
body: plane.numberOfSeats - passengers->size()
```

- ¿Cómo se expresa el hecho de que en ningún vuelo puede haber más pasajeros que asientos tiene el avión?
- Restricción OCL:

Context Flight

Inv: `passengers->size() <= plane.numberOfSeats`

Ejemplos



Reglas adicionales:

1. Una persona puede tener una hipoteca sobre una casa sólo si es el propietario.
2. La fecha de inicio de cada hipoteca ha de ser menor que la de final.
3. El número de la seguridad social de cada persona ha de ser único.
4. Sólo es posible contratar una nueva hipoteca si el salario de la persona es suficiente.
5. Sólo es posible contratar una nueva hipoteca si el valor de la casa aval es suficiente.

Ejemplos

Las restricciones OCL se escriben en el contexto de una instancia de un tipo específico.

Context Hipoteca

Inv: aval.propietario = contratante

Context Hipoteca

Inv: fechaInicio < fechaFin

self hace referencia a la instancia del contexto.

Context Persona

Inv: Persona::allInstances() -> isUnique(numSegSoc)

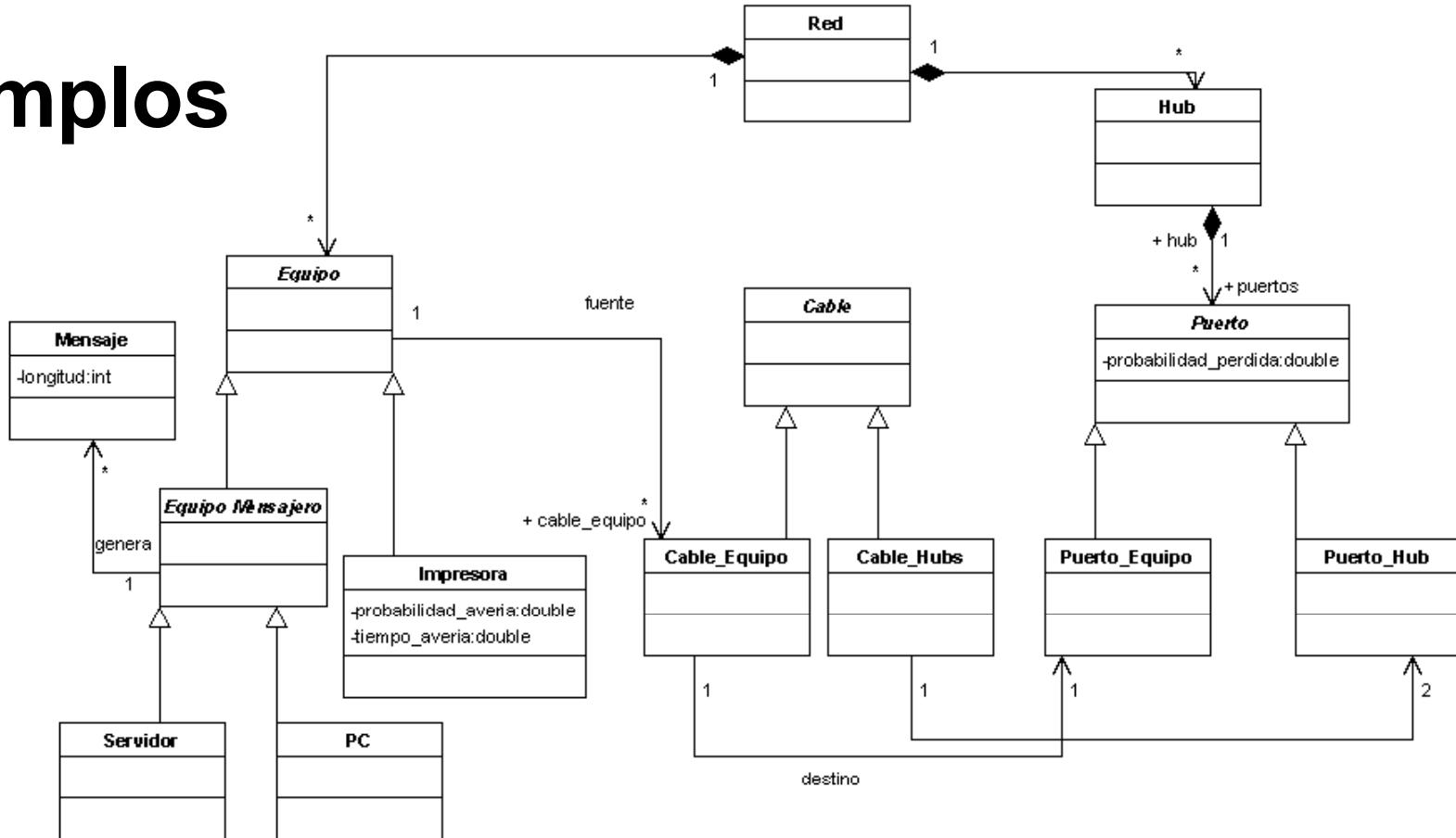
Context Persona::contratarHipoteca(sum: Dinero, aval: Casa)

pre: self.hipotecas.mensual->sum() + sum <= self.sueldo * 0.70

Context Persona::contratarHipoteca(sum: Dinero, aval: Casa)

pre: aval.valor >= aval.hipotecas.principal->sum()

Ejemplos



“Los PCs pueden conectarse con un único Hub, los servidores con uno o varios”

Context PC

Inv: `cable_equipo->size() = 1`

Context Servidor

Inv: `cable_equipo->size() >= 1`

Ejercicio

“Un Hub no puede conectarse consigo mismo a través de un puerto”

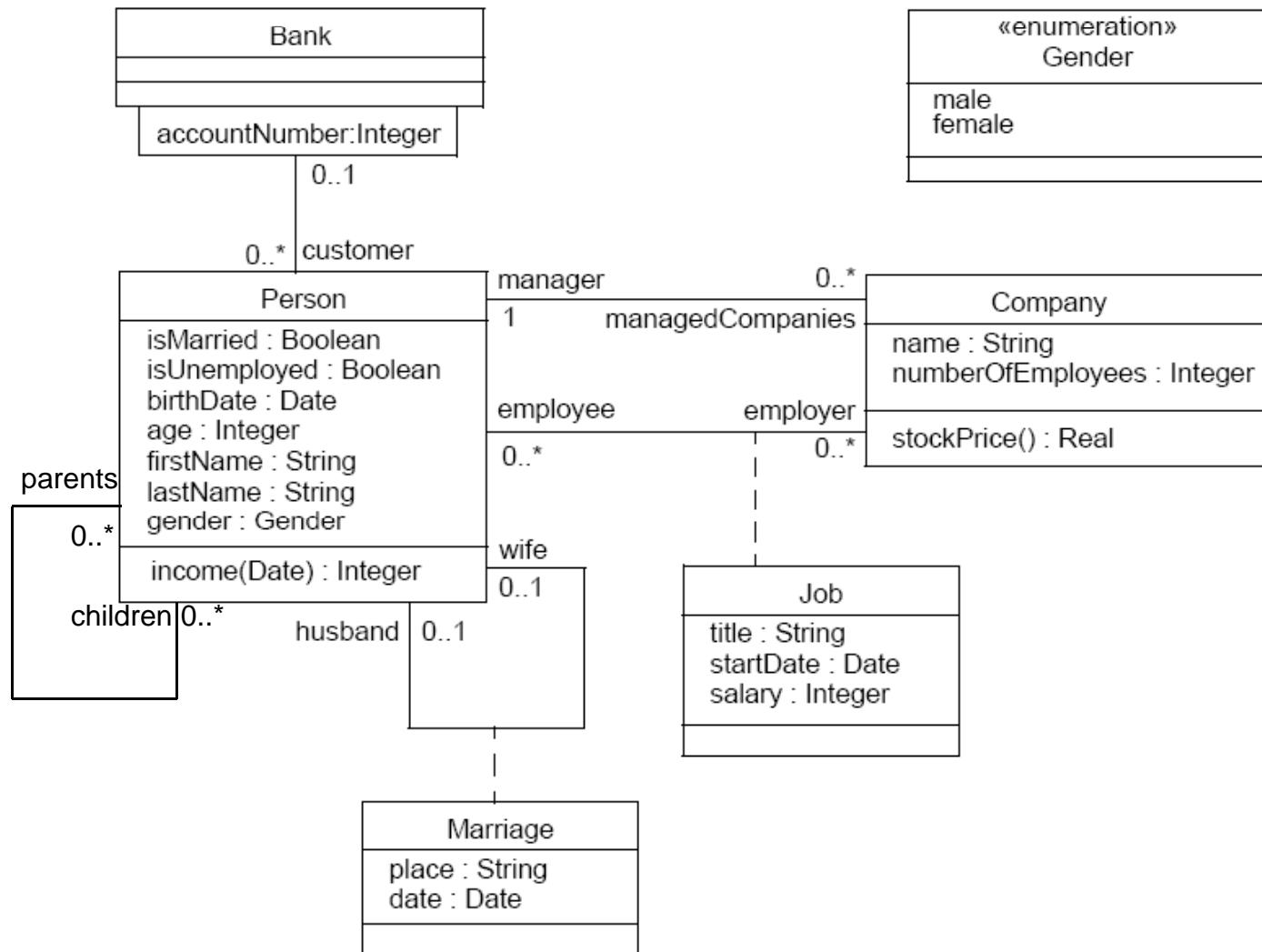
Context Cable_Hubs

Inv: Puerto_Hub.hub->asset()->size() = 2

¿Dónde usar OCL?

- Clases:
 - **Invariantes.** Expresión OCL de tipo booleano, la expresión debe ser true para cada instancia de la clase en todo momento de la ejecución.
- Operaciones:
 - **Pre-condición.** Condición que debe ser verdadera para ejecutar la operación en una determinada instancia.
 - **Post-condición.** Condición que debe ser verdadera al terminar una operación.
 - **Body.** Especificación del cuerpo de una operación de tipo query.
- Atributos y finales de asociación:
 - **Valor inicial.** expresión para dar el valor inicial a un atributo o final de asociación. Se evalua al crear la instancia.
 - **Valor derivado.**
- Transiciones de máquinas de estados:
 - **Guarda.**

Ejemplos



Ejemplos

- Especificación del valor inicial y derivado de atributos/association ends:

```
context Person::income : Integer
init: parents.income->sum() * 1% -- pocket allowance
derived:if self.age < 18
    then parents.income->sum() * 1% -- pocket allowance
    else job.salary -- income from regular job
endif
```

- Subexpresiones (let):

```
context Person inv:
let income : Integer = self.job.salary->sum() in
if isUnemployed then
income < 100
else
income >= 100
endif
```

Ejemplos. Colecciones.

- Tipos: Set, OrderedSet, Bag, Sequence.
- Operaciones de bucle con colecciones:

select(expr): selecciona los elementos que cumplan una condición.

colección->select(expresión_lógica)

colección->select(v | expresión_lógica_con_v)

colección->select(v : Type | expresión_lógica_con_v)

context Company inv:

```
self.employee->select(age < 25)->notEmpty()
```

context Company inv:

```
self.employee->select(gender=female)->notEmpty()
```

Ejemplos. Colecciones.

collect(expr): devuelve la colección que resulta de evaluar expr para cada elemento de la colección fuente.

```
self.employee->collect( birthDate )->asSet()
```

forAll(expr): devuelve verdadero si expr es verdadero en cada elemento de la colección.

```
colección->forAll( expresion-logica )
colección->forAll( v | expresion-logica-con-v )
colección->forAll( v : Type | expresion-logica-con-v )
```

```
context Company
```

```
inv: self.employee->forAll( isUnemployed = False )
inv: self.employee->forAll( p | p.isUnemployed = False )
inv: self.employee->forAll( p : Person | p.isUnemployed = False )
```

Ejemplos. Colecciones.

exists(expr): devuelve true si al menos hay un elemento en la colección para el que expr es verdadera.

```
colección->exists( expresion-logica )
```

```
colección->exists( v | expresion-logica-con-v )
```

```
colección->exists( v : Type | expresion-logica-con-v )
```

```
context Company
```

```
inv: self.employee->exists( age > 50 )
```

```
inv: self.employee->exists( p | p.age > 50 )
```

```
inv: self.employee->exists( p : Person | p.age > 50 )
```

iterate(...): itera sobre todos los elementos de una colección.

```
colección->iterate( elem : Type; acc : Type = <expresion> | expresion-logica-con-elem-y-acc )
```

```
collection->collect(x : T | x.property)
```

```
collection->iterate(x : T; acc : T2 = Bag{} | acc->including(x.property))
```

añade un elemento a una colección

Colecciones. Otras Operaciones.

- Otras operaciones de bucle:
 - source->any(iterator|body)
 - source->select(iterator|body)->asSequence()->first()
 - source->collectNested(iterators|body)
 - Bag de elementos que resultan de aplicar body a cada elemento de source.
 - source->isUnique(iterators|body)
 - True si body se evalua a un valor diferente para cada elemento de source.
 - source->one(expr)
 - Devuelve true si existe exactamente un elemento de source que cumple la condición.
 - source->reject(expr)
 - Devuelve una colección con los elementos de source que no cumplen la condición.
 - source->sortedBy(expr)
 - Ordena source, resulta en un OrderedSet

Indice

- Diagramas de Casos de Uso.
- Diagramas de Estructura.
- Diagramas de Comportamiento.
- OCL.
- **Herramientas.**
- Ejemplos.
- Bibliografía.

Herramientas, I

- Dibujo de diagramas
 - Soporte a la corrección de los diagramas en base a su semántica
 - Apoyo para simplificar la facilidad de comprensión de los diagramas (layout, etc.)
- Archivo de información
 - Comprobación de inconsistencias
 - Detección de trabajo a realizar y mejoras
 - Generación de informes
 - Soporte a la reutilización
 - Soporte al rediseño (refactorings).

Herramientas, II

- Soporte a la navegación
 - Vistas compuestas
 - Elaboración de conexiones entre información relacionada
 - Búsqueda
 - Visión con diferentes niveles de granularidad (expansión y contracción de partes de la vista)
 - Filtros
 - Operaciones sobre componentes de la vista

Herramientas, III

- Soporte al trabajo multiusuario
 - Bloqueo de información
 - Trabajo colaborativo
- Generación de código
 - Esqueletos con información estática (clases)
 - Generación a partir de diagramas de comportamiento (máquinas de estados).
 - Integración con lenguajes especiales (SQL, ...)
 - Desarrollo Dirigido por Modelos (MDA,
<http://www.omg.org/mda/>)
- Ingeniería inversa
 - Construcción de un modelo de diseño a partir de código
 - Diseño iterativo: incorporación al modelo de detalles implementados

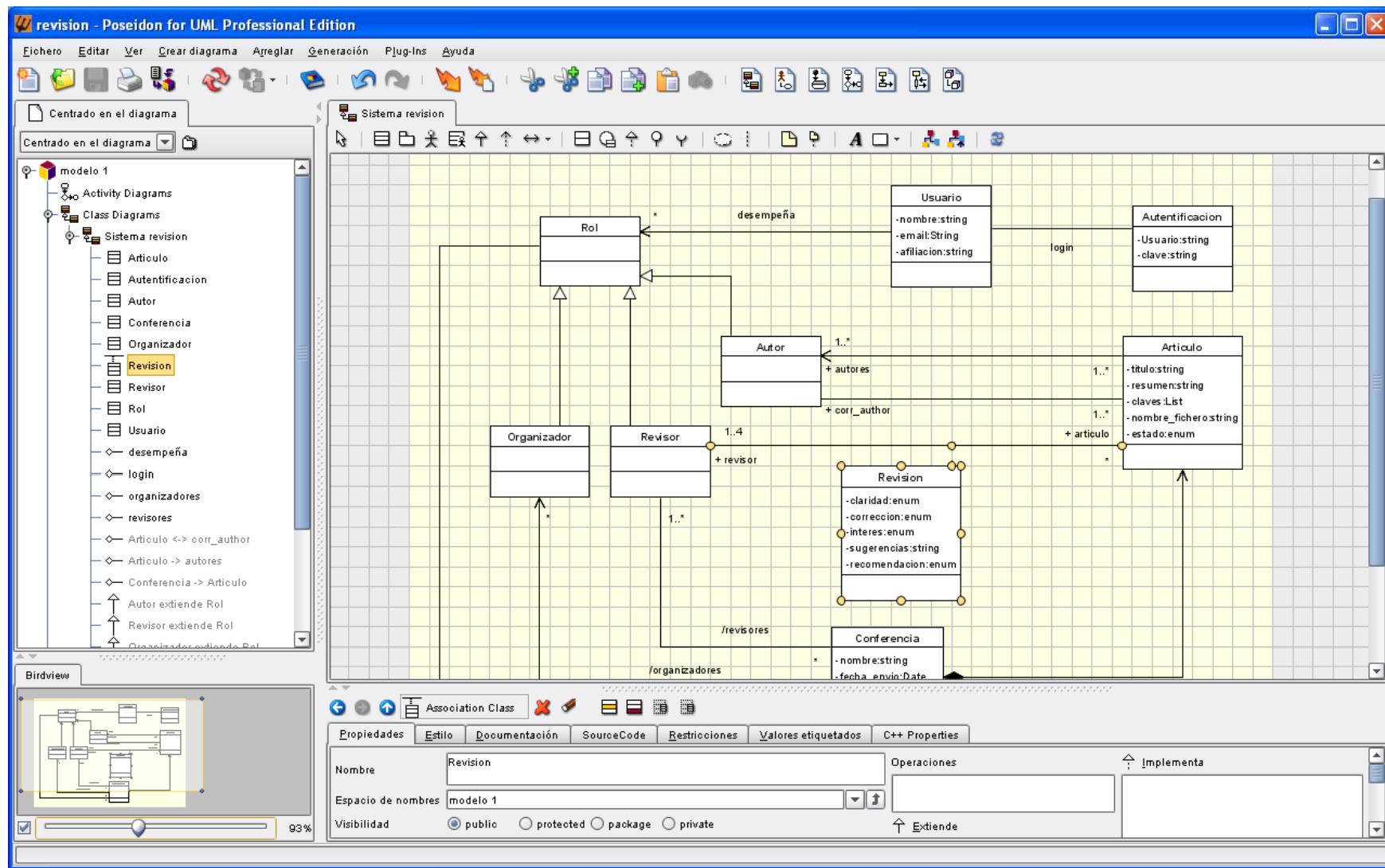
Herramientas, IV

- Integración con otras herramientas
 - Entorno de desarrollo (tendencia a confluir)
 - Configuración del sistema y control de versiones
 - Herramientas de documentación
 - Herramientas de prueba de software
 - Herramientas de construcción de interfaces de usuario
 - Herramientas de especificación de requisitos
 - Herramientas de gestión de proyectos y soporte al proceso de diseño y desarrollo

Herramientas, V

- Distintos niveles de abstracción
- Intercambio de información
 - Especificación OMG de representación de modelos UML en XMI (XML Metadata Interchange)

Poseidon for UML



Indice

- Diagramas de Casos de Uso.
- Diagramas de Estructura.
- Diagramas de Comportamiento.
- OCL.
- Herramientas.
- **Ejemplos.**
- Bibliografía.

Ejemplo de Análisis

Aplicación para una Galería de Arte

Diagrama de casos de uso inicial

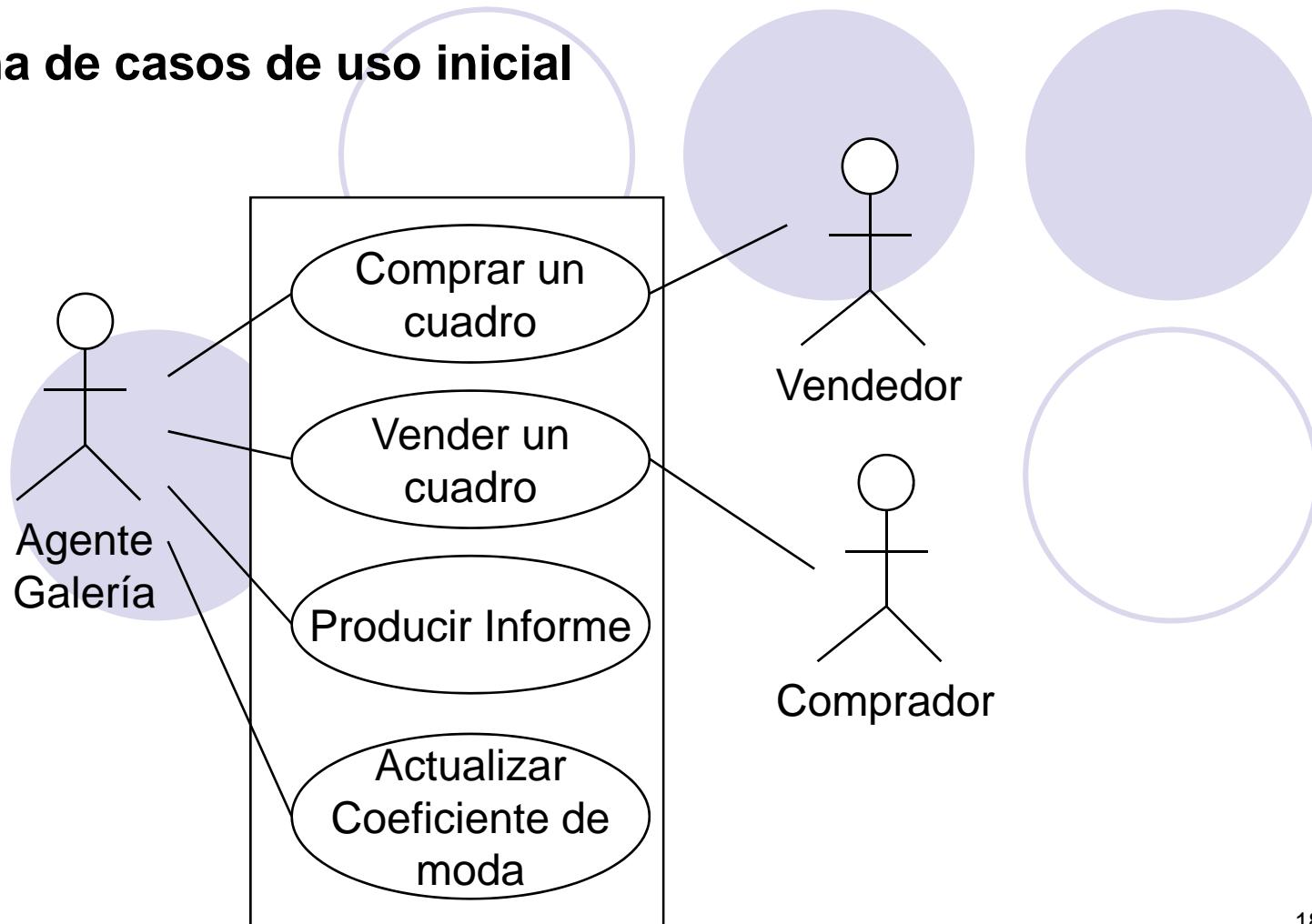
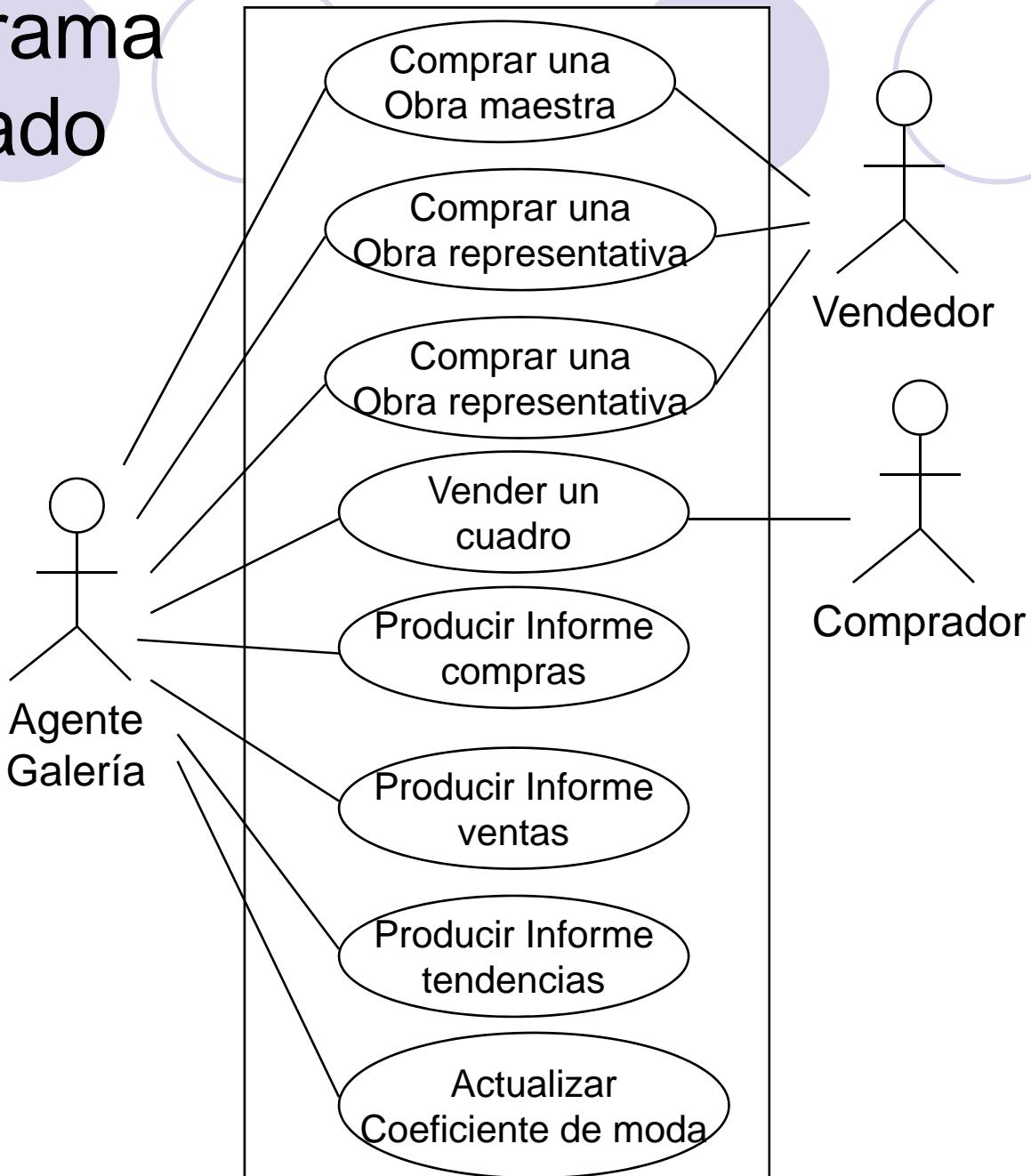


Diagrama refinado



Caso de uso: comprar una obra maestra

Actores primarios:

Agente Galería, vendedor

Interesados y Objetivos:

- **Agente**: quiere obtener una recomendación lo más acertada posible del precio máximo recomendado de manera rápida.
- **Vendedor**: quiere vender el cuadro a un precio razonable de manera rápida.

Precondiciones:

El agente ha entrado en la aplicación.

Garantía de éxito (post-condiciones):

Se registra la venta.

Escenario Principal de Éxito:

1. El agente introduce la descripción del cuadro.
2. El sistema busca el cuadro más parecido del mismo autor.
3. El sistema presenta el precio recomendado.
4. El agente hace una propuesta por debajo del precio recomendado, y el vendedor acepta la oferta.
5. El agente introduce información de la venta.

Extensiones:

- 2a. No hay ningún cuadro parecido del mismo autor, así que el sistema no presenta una recomendación.
- 4a. El vendedor no acepta la oferta y la venta no se produce.

Diagrama de clases inicial

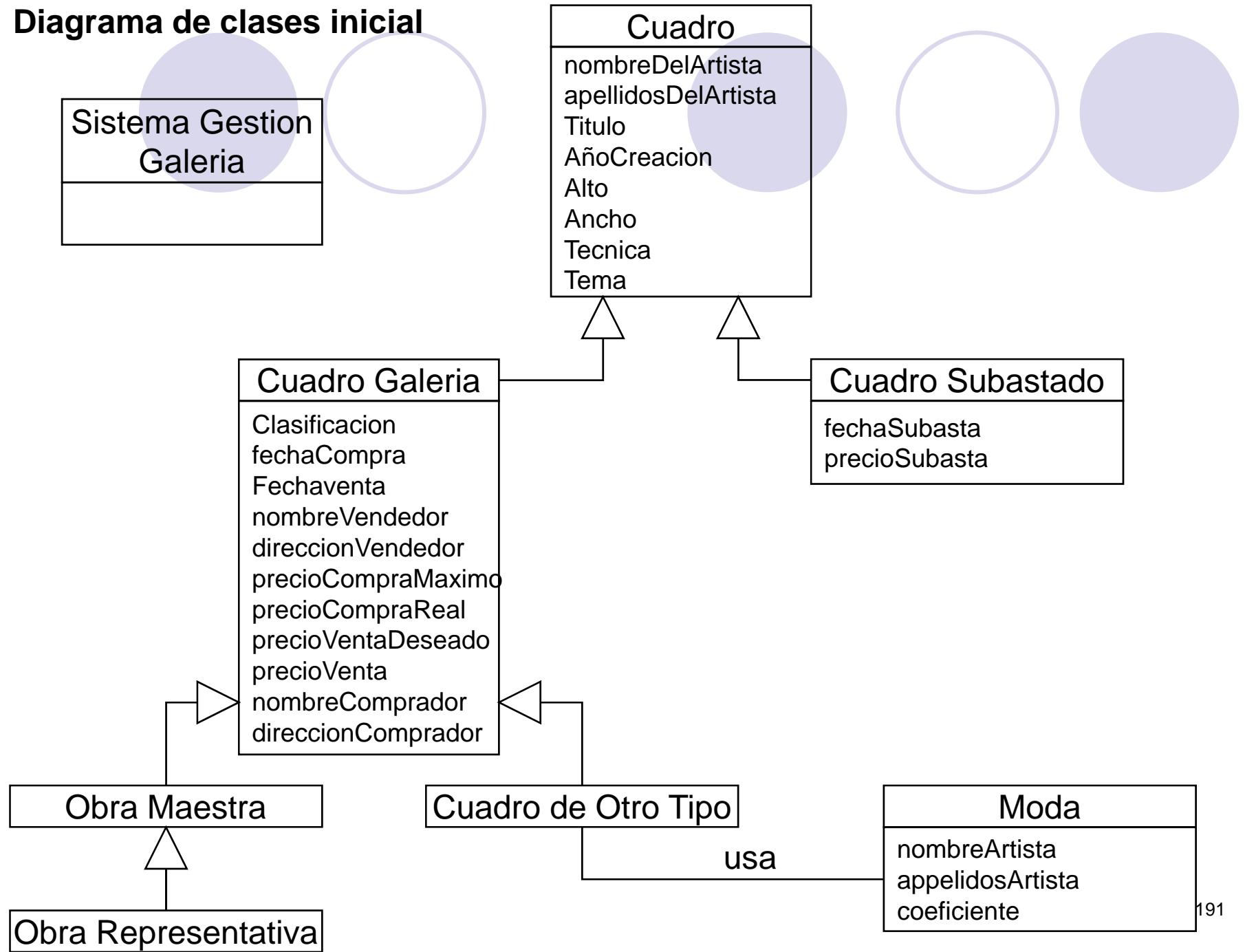
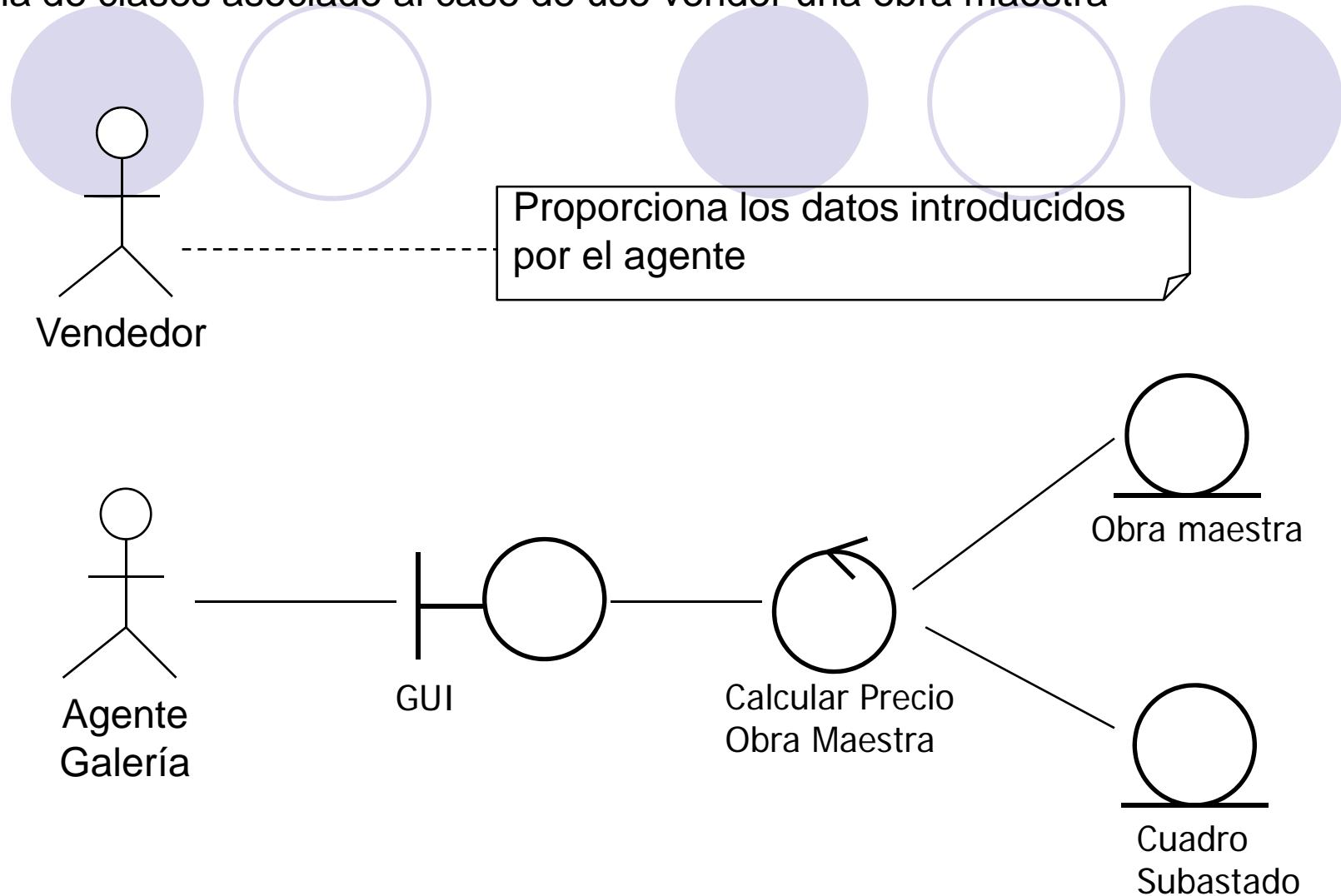
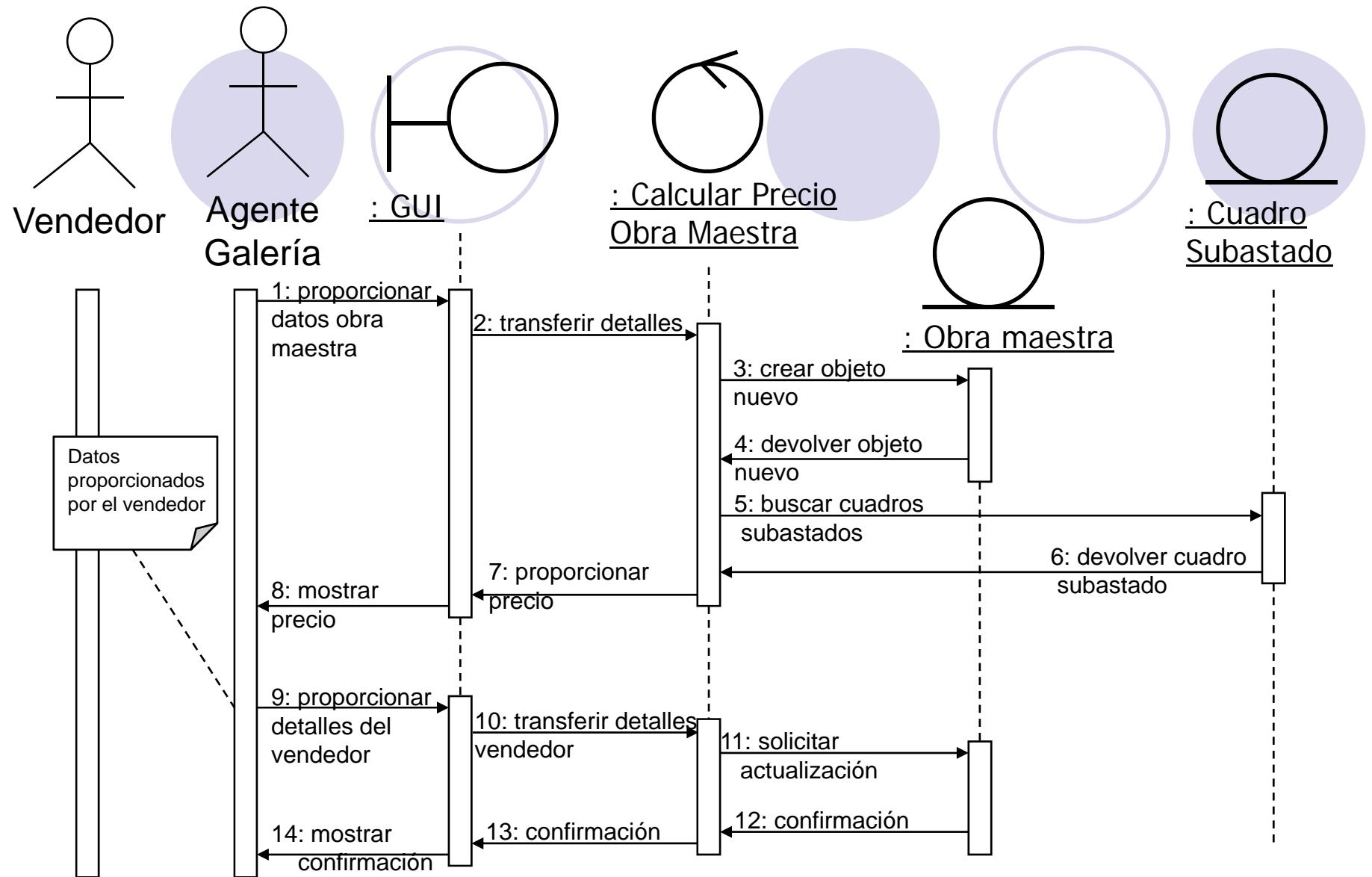
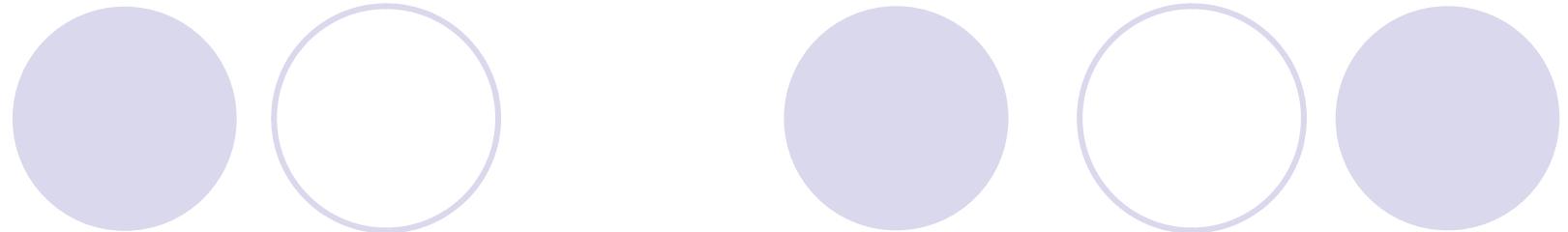


Diagrama de clases asociado al caso de uso vender una obra maestra



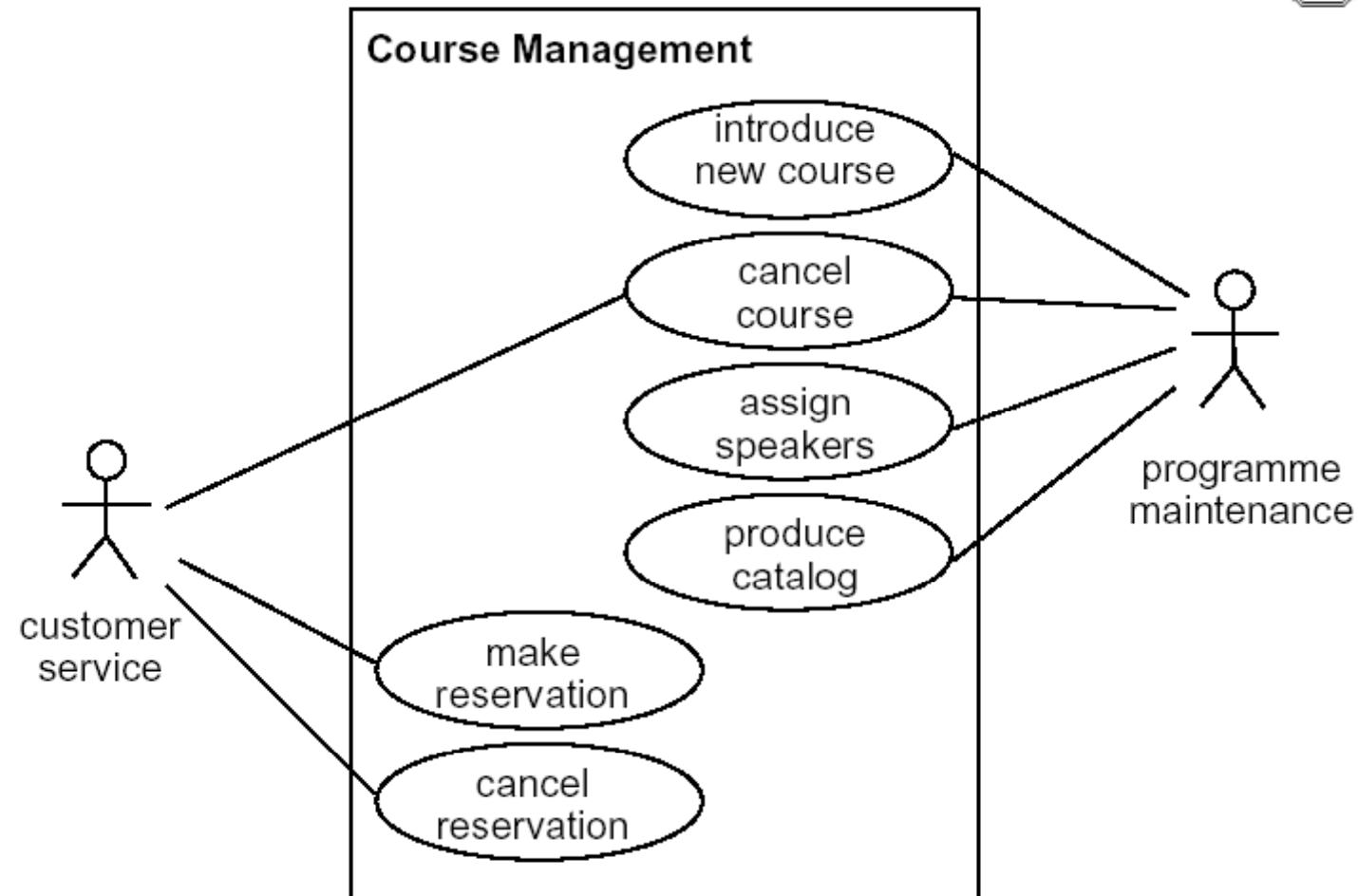




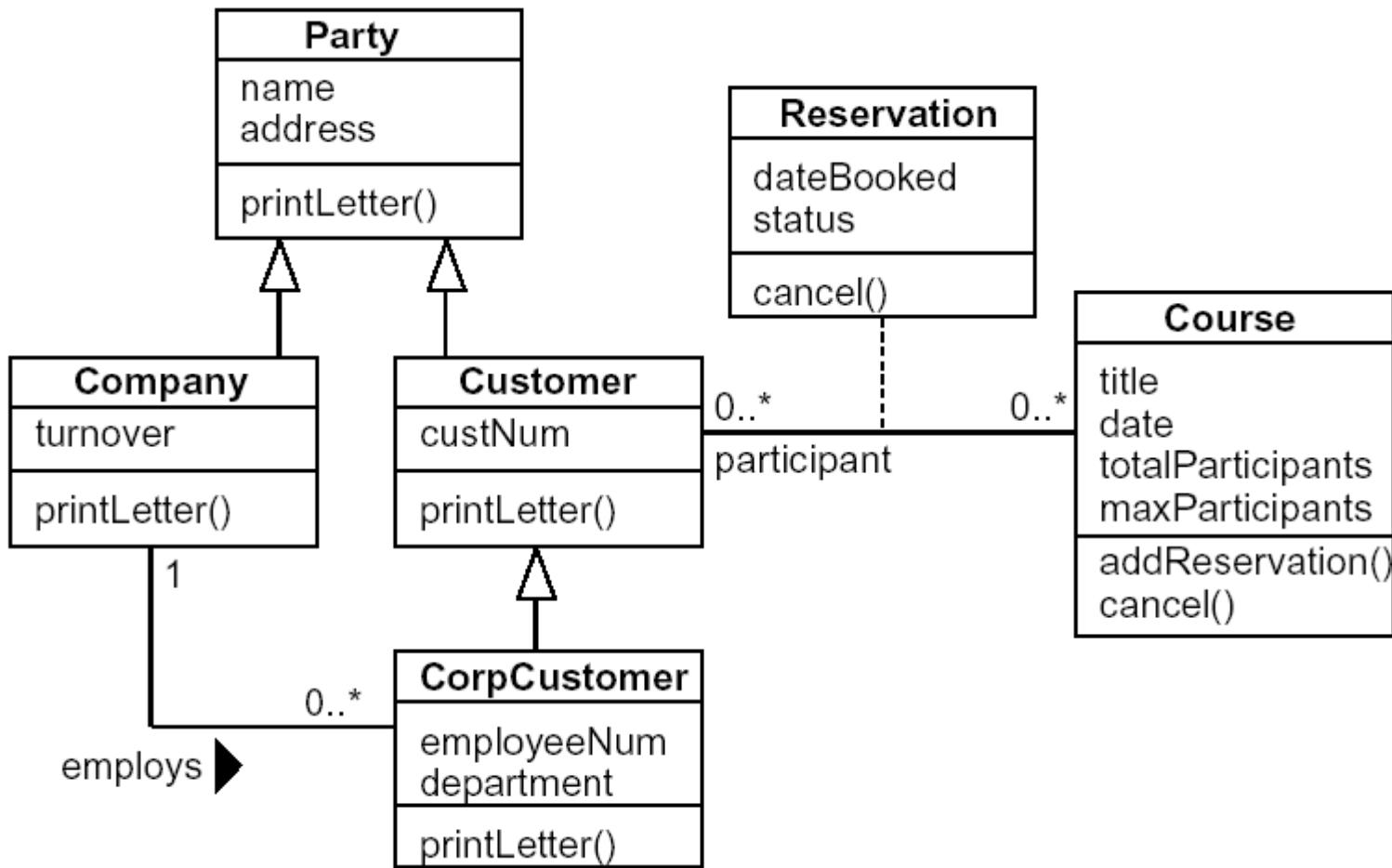
Un ejemplo completo para todos los diagramas

Un sistema de gestión de cursos impartidos para un conjunto de clientes. Algunos de esos clientes pueden pertenecer a empresas.

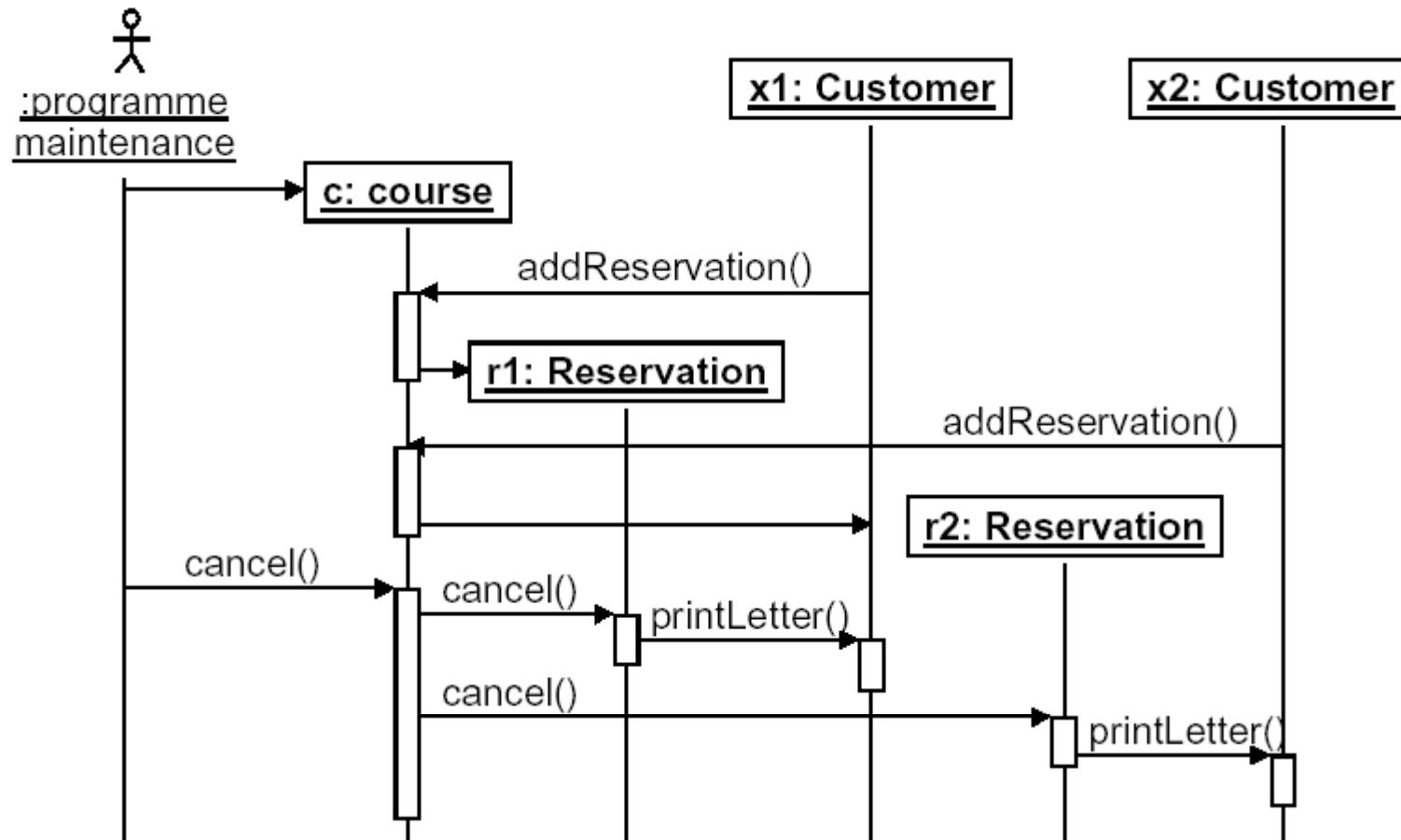
Use Case Diagram



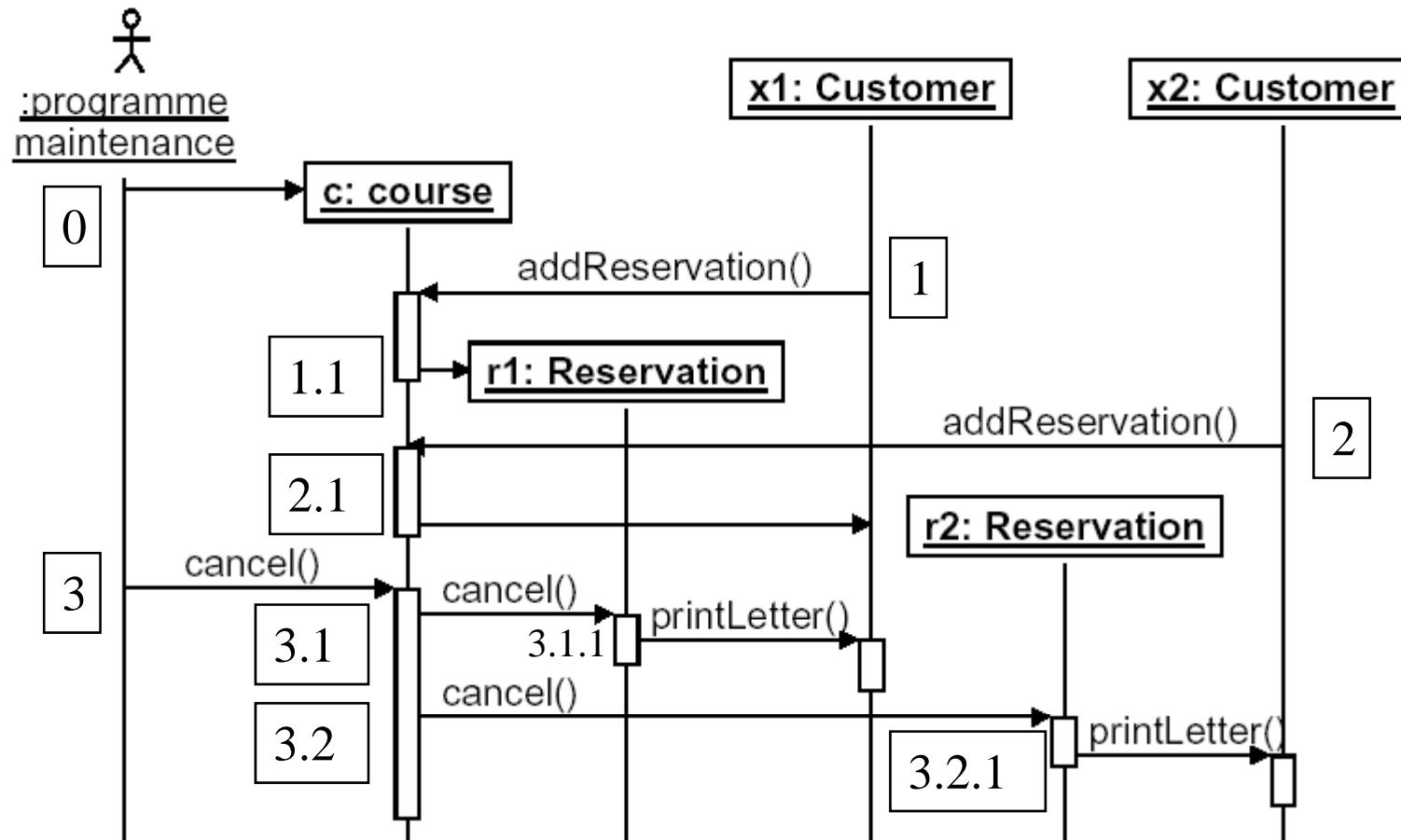
Static Structure Diagram: Class Diagram



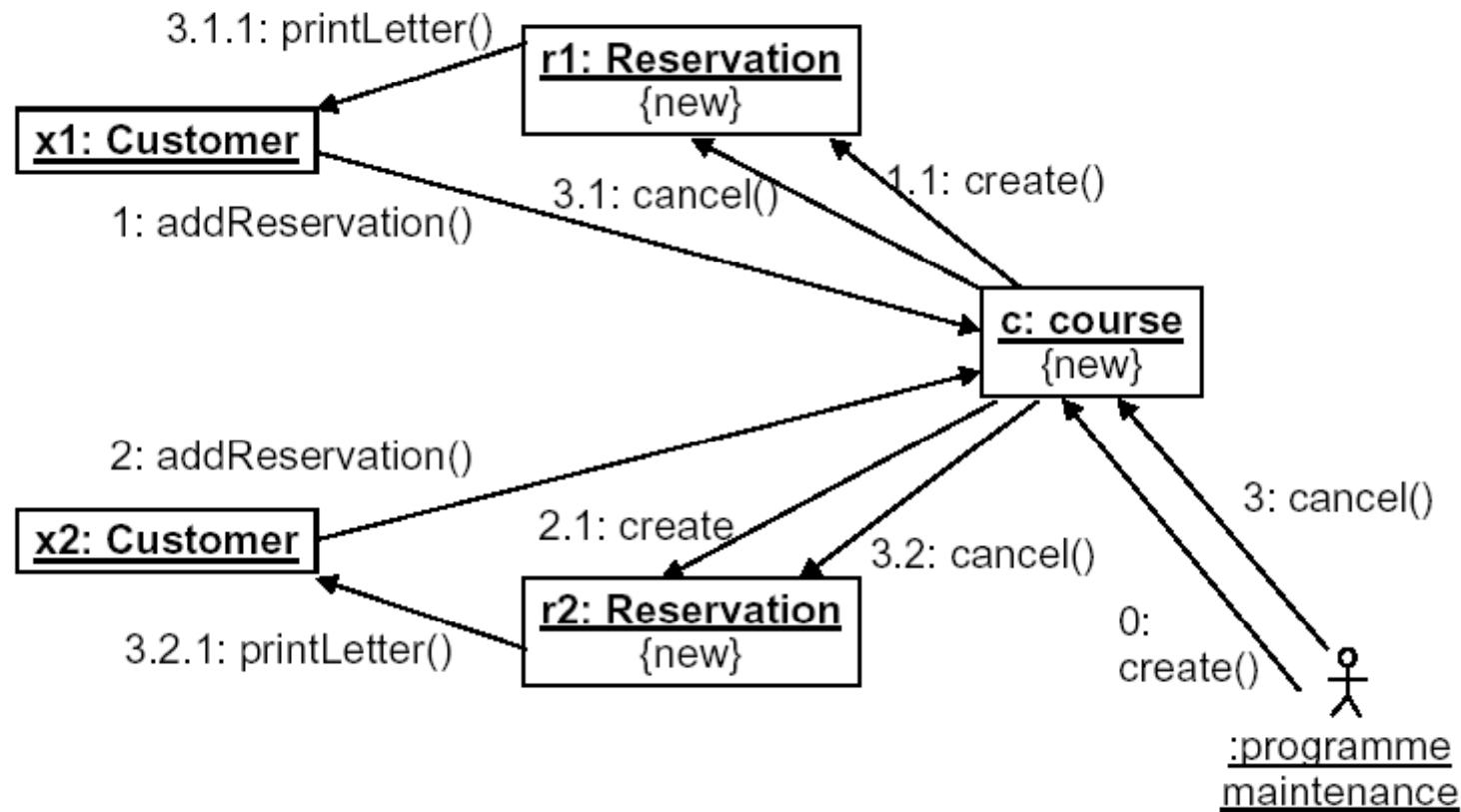
Interaction Diagram: Sequence Diagram



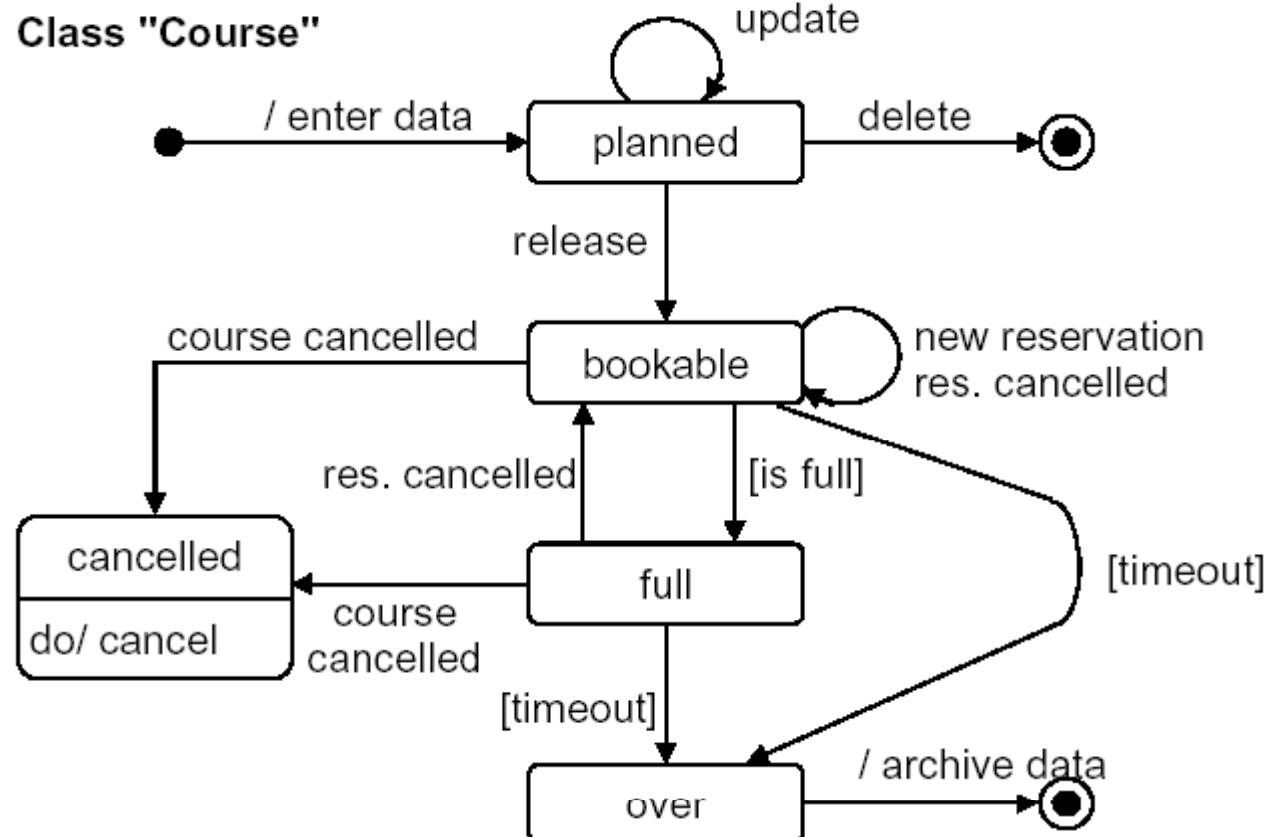
Interaction Diagram: Sequence Diagram



Interaction Diagram: Collaboration Diagram

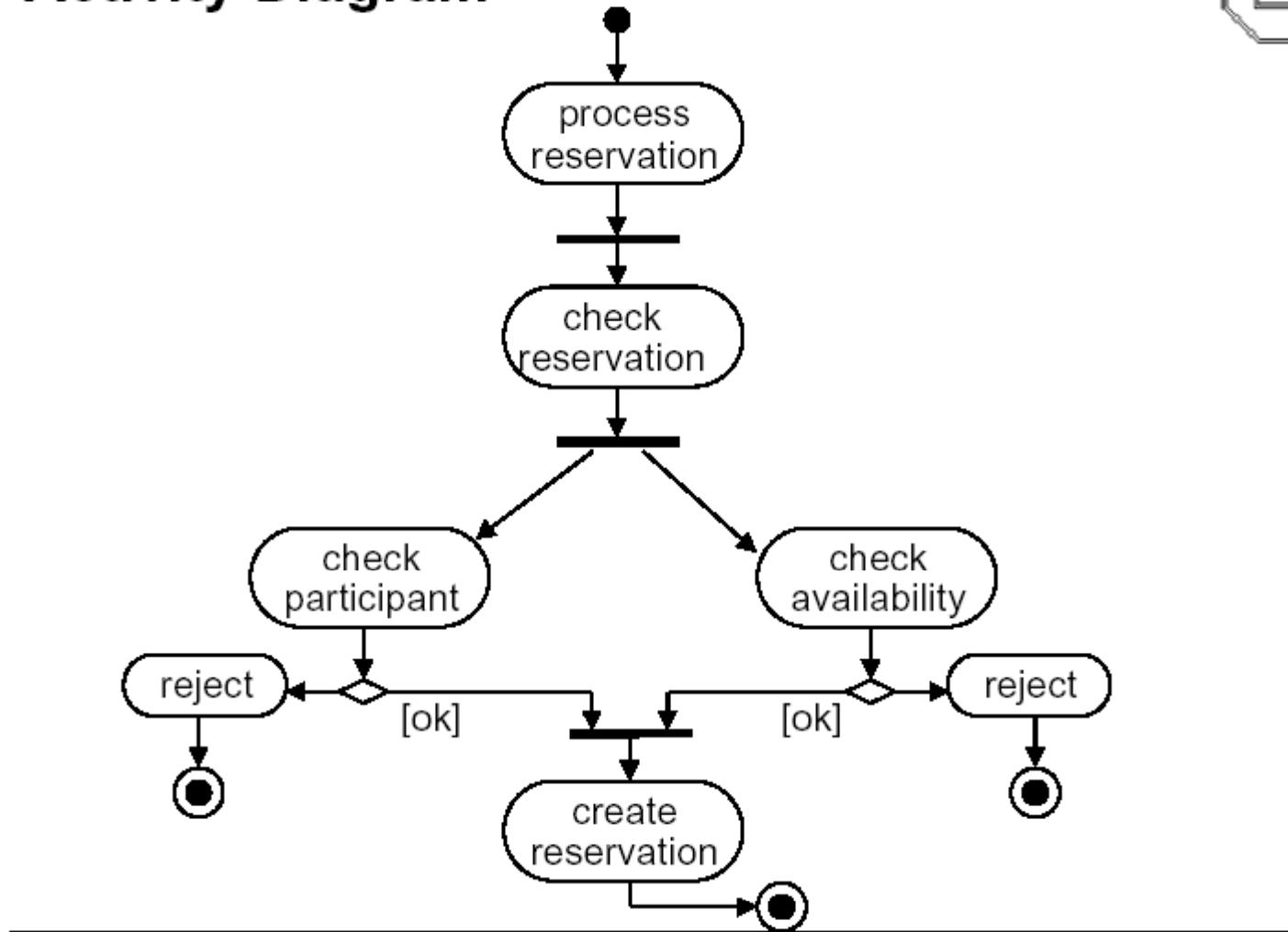


Statechart Diagram

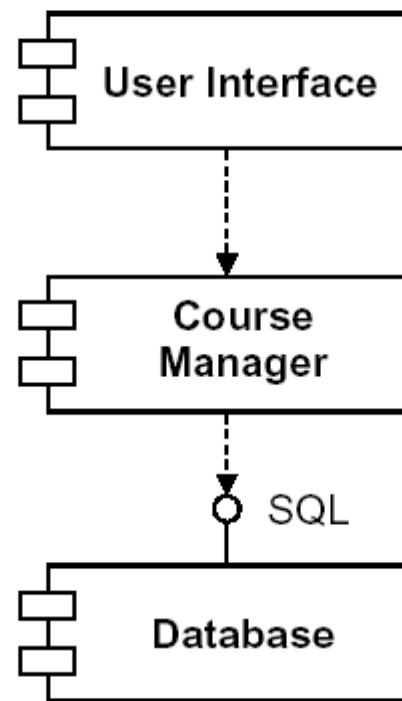


is_full: totalParticipants = maxParticipants
timeout: today >= date

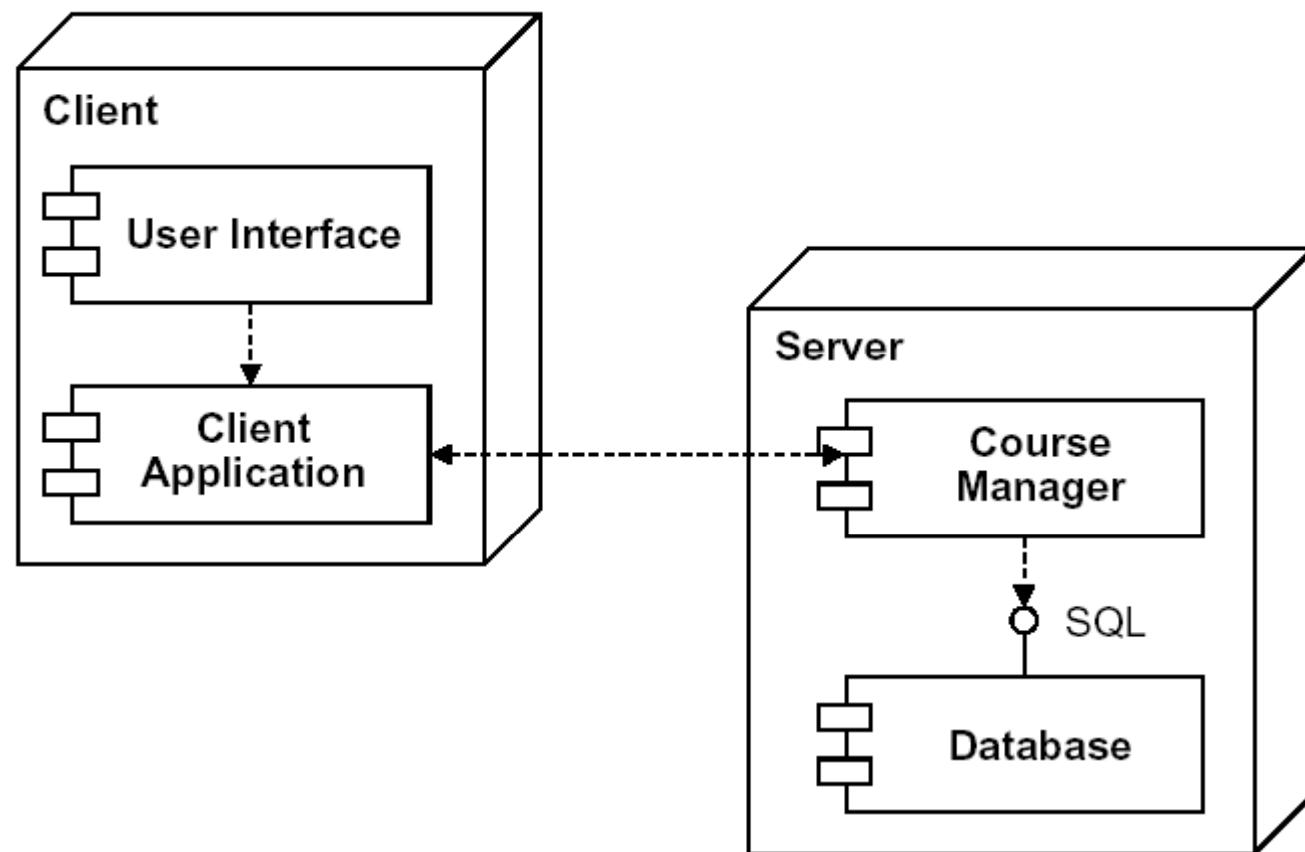
Activity Diagram



Implementation Diagram: Component Diagram



Implementation Diagram: Deployment Diagram



Indice

- Diagramas de Casos de Uso.
- Diagramas de Estructura.
- Diagramas de Comportamiento.
- OCL.
- Herramientas.
- Ejemplos
- **Bibliografía.**

Bibliografía: UML

- Dan Pilone; Neil Pitman. “*UML 2.0 in a Nutshell*”. O'Reilly, 2005.
- Martin Fowler. “*UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition*”. Addison Wesley, 2003.
- Web de la OMG sobre UML: <http://www.uml.org>
- Perdita Stevens, Rob Pooley. “*Utilización de UML en Ingeniería del Software con Objetos y Componentes*”. Addison Wesley, 2002.
- Grady Booch, James Rumbaugh, Ivar Jacobson. “*The Unified Modeling Language User Guide*”. Addison-Wesley, 1999.
- Eriksson, H. E., Penker, M., Lyons, B., Fado, D. “*UML 2 Toolkit*”. OMG Press, Wiley. 2004.
- Craig Larman. “*Applying UML and Patterns*”. Prentice Hall. 2002.

OCL

- Warmer, Kleppe. “*The Object Constraint Language 2nd Edition. Getting your Models Ready for MDA*”. Addison-Wesley. 2003.
- Especificación de OCL 2.0: <http://www.omg.org/docs/ptc/03-10-14.pdf>

Bibliografía: Statecharts

- Harel D. “*On Visual Formalisms*”. Communications of the ACM. Vol 31, No. 5. Pp.: 514-530. Mayo 1988.
- Harel D., Naamad A. “*The STATEMATE Semantics of Statecharts*”. ACM Transactions on Software Engineering and Methodology, Vol. 5, No. 4, Oct. 1996, pp.: 293-233.
- David Harel and Eran Gery. Executable object modeling with statecharts. IEEE Computer, pages 31-42, 1997.