



Análisis – Modelado dinámico

Índice

- 1. Introducción al modelado dinámico
- 2. Interacciones o Colaboraciones
 - ❖ 2.1. Diagramas de Comunicación
 - ❖ 2.2. Diagramas de Secuencia
- 3. Diagramas de Estados
- 4. Diagramas de Actividades

*Notación conforme con la
especificación 2.5.1 de UML*



1. Introducción al modelado dinámico

- En el *modelo de objetos* se representó la estructura del sistema (clases y sus relaciones)
- En el *modelo dinámico* estudiamos:
 - ❖ Cómo evolucionan los objetos a lo largo del tiempo
→ **diagrama de estados**
 - ❖ Cómo colaboran los objetos entre sí para llevar a cabo los servicios que ofrece el sistema →
diagramas de interacción

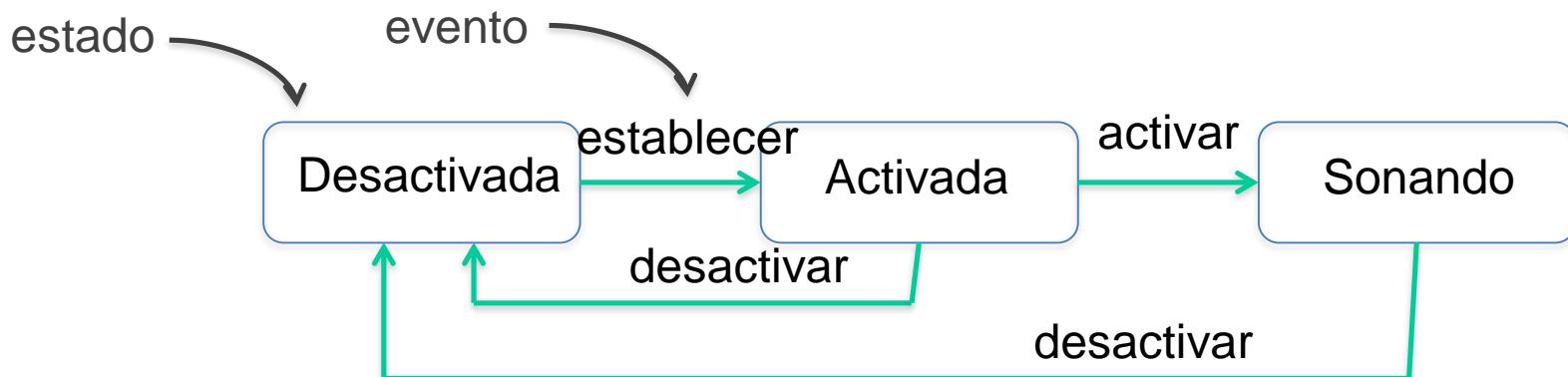
Modelo dinámico ...

- El *control*/es aquella parte del sistema que describe las secuencias de *operaciones* que se producen en respuesta a *eventos*
- El modelo dinámico estudia el orden en que se llevan a cabo los eventos y las operaciones en el sistema
- Además se especifica qué hacen las operaciones →
diagrama de actividades

Ejemplo

- Los conceptos más importantes del modelo dinámico son:
 - ❖ Evento, Estado y Mensaje

Ejemplo de eventos y estados en el diagrama de estados de un objeto alarma:



Eventos

- ❑ Un evento es un *estímulo* que se envía a un objeto bien por parte de otro objeto bien por parte de un actor
- ❑ Por tanto el evento ocurre en el entorno del sistema y consideraremos que no tiene duración
- ❑ Ejemplos de eventos:
 - ❖ El vuelo 123 sale para Madrid
 - ❖ Pulsar el botón izquierdo del ratón (posición)

Eventos ...

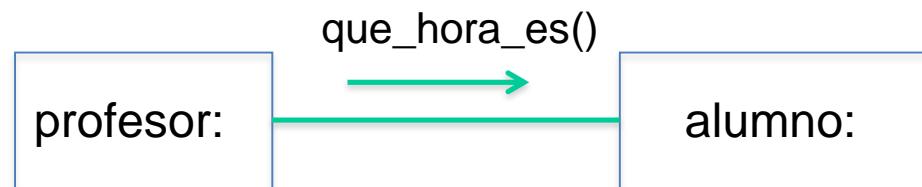
- Dos eventos que no tienen relación causal son concurrentes, no tienen efecto entre sí
- Al modelar un sistema no intentamos establecer un orden entre sucesos concurrentes

Mensaje

- Es la unidad de **comunicación** entre objetos
- Es el proceso de presentar a un objeto una solicitud para que realice una acción específica
- Contiene:
 - ❖ Objeto emisor
 - ❖ Objeto receptor
 - ❖ Nombre de la *operación* invocada
 - ❖ Posible lista de argumentos

Mensaje ...

Ejemplo de envío de mensajes en un diagrama de comunicación:



Mensaje ...

- ❑ Para que un objeto responda a un mensaje, la operación debe estar en su interface pública
- ❑ Al conjunto de mensajes a los que puede responder un objeto le llamamos *comportamiento*
- ❑ El receptor realiza la operación en la forma que sólo él conoce (puede ser cualquiera, con tal de que haga lo solicitado)
- ❑ El emisor no puede decir cómo se lleva a cabo la operación

Mensaje ...

- El *método* es el algoritmo que se ejecuta en respuesta al mensaje
- El mensaje debería ser el único *acoplamiento* existente entre los objetos/clases
- El mensaje garantiza la delegación de tareas
- El mensaje adquiere mayor importancia con el polimorfismo y el enlace dinámico

Estado

- ❑ Lo hemos definido como una abstracción de los valores de los atributos y de los enlaces de un objeto
- ❑ La respuesta a un evento depende del estado del objeto que lo recibe:
 - ❖ Puede implicar un cambio de estado en el receptor
 - ❖ Puede implicar que el receptor envíe otro evento (a un tercer objeto o al emisor)

Estado ...

- ❑ Ejemplo de diagrama de estados para objetos de tipo Cuenta Bancaria:

Clase: Cuenta Bancaria

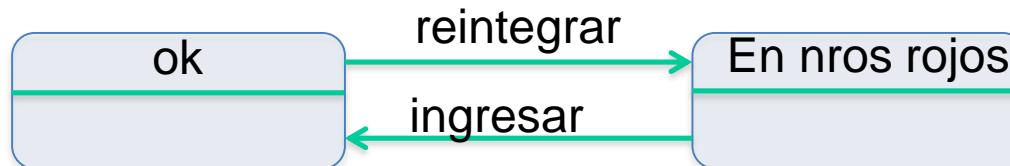
atributos: saldo, fecha de apertura, etc

Estados:

ok saldo ≥ 0

En números rojos saldo < 0

CuentaBancaria
saldo
fechaApertura
...
reintegrar
ingresar



Estado ...

- Un estado corresponde al intervalo entre dos eventos recibidos por el objeto
- Los eventos son inmediatos, los estados tienen duración
- Normalmente un evento separa dos estados y un estado separa dos eventos

2. Interacciones o Colaboraciones

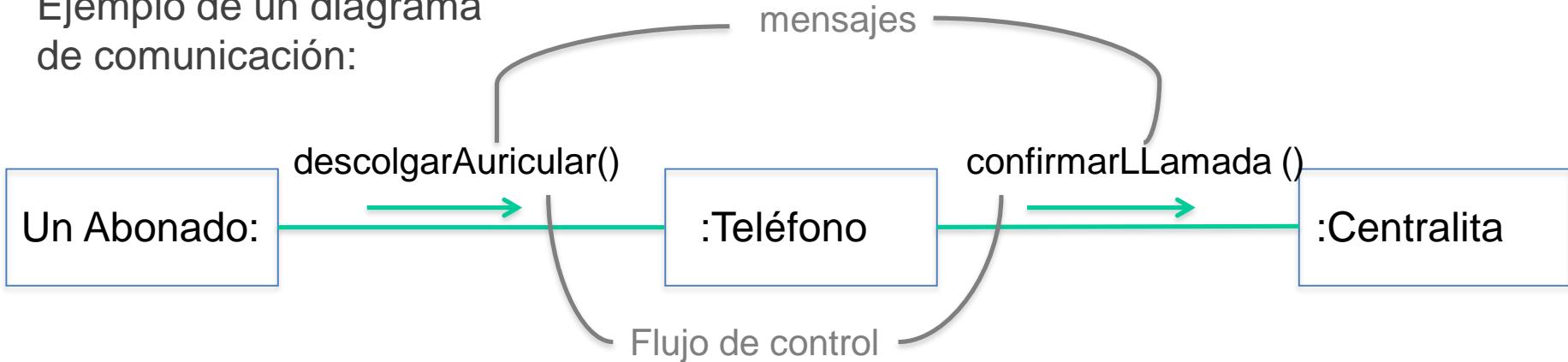
- ❑ Los objetos **colaboran** en la consecución de un fin
→ realizar las funciones de la aplicación
- ❑ La colaboración o **interacción** se realiza comunicándose por medio de **mensajes**
- ❑ Un mensaje se envía para solicitar a un objeto un **servicio**

Colaboraciones

- ❑ Una interacción o colaboración es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos dentro de un *contexto* para lograr un propósito
- ❑ Los objetos interactúan para llevar a cabo los servicios ofrecidos por el sistema. Las interacciones muestran cómo se comunican los objetos.

Ejemplo de colaboración

Ejemplo de un diagrama de comunicación:



- Un diagrama de objetos se puede ver como una representación estática de una interacción
- La interacción va más allá porque introduce la secuencia de mensajes que fluyen a través de los enlaces

Colaboraciones: Contexto

□ El contexto puede ser especificar:

- ❖ el sistema completo
- ❖ un subsistema
- ❖ una clase
- ❖ una operación
- ❖ un caso de uso
- ❖ un módulo

Contexto ...

- Modelar interacciones es útil tanto en Análisis como en Diseño
 - ❖ Análisis: entender de manera común cómo se desarrolla una comunicación en el sistema
 - ❖ **Diseño:** Definir una forma/protocolo de comunicación

Elementos de una colaboración/interacción

- Objetos y roles
- Enlaces y conectores
- Mensajes

Objetos y roles

- ❑ Los objetos pueden ser concretos o prototípicos (normalmente prototípicos)
- ❑ Los objetos prototípicos se llaman *roles*
- ❑ Podemos distinguir varios tipos de objetos estudiando su *comportamiento* en la colaboración:
 - ❖ Clientes o actores
 - ❖ Servidores
 - ❖ Agentes

Tipos de objetos (I)

□ Clientes

- ❖ Son el origen de la interacción. Solicitan a los servidores un servicio. Pasan el testigo a otros objetos

□ Servidores:

- ❖ No son el origen de la interacción. Ofrecen sus servicios a la sociedad de objetos. Reciben los mensajes



Tipos de objetos (II)

□ Agentes:

- ❖ Son clientes y servidores a la vez. Actúan por iniciativa propia o por una solicitud externa
- ❖ Permiten que un cliente se comunique con un servidor al que no conoce
- ❖ Son la base de la *delegación*

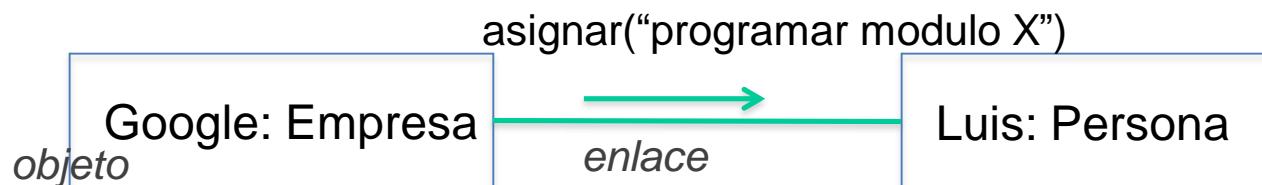
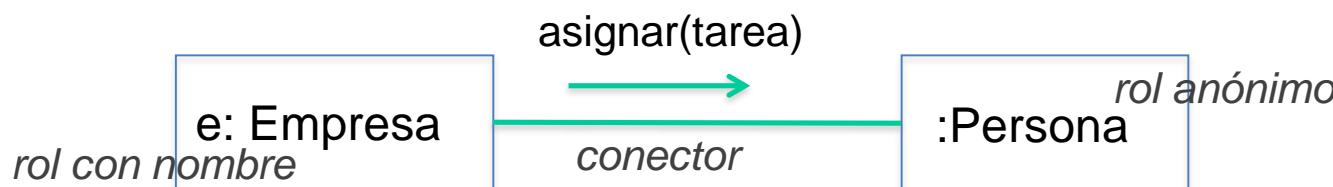
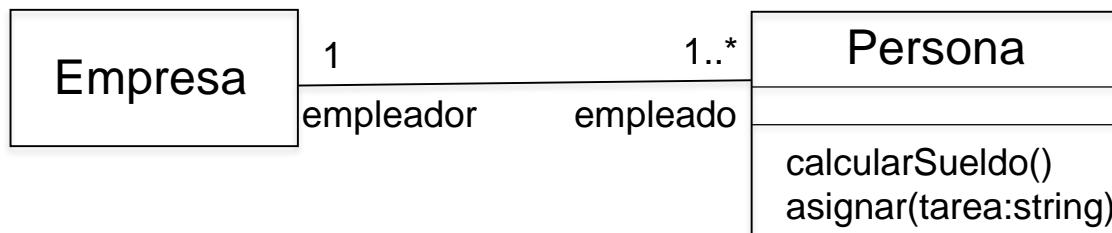


Enlaces y conectores (I)

- Como en el diagrama de objetos un enlace es una instancia de una asociación → si hay una asociación entre dos clases entonces puede haber un enlace entre sus instancias
- Siempre que hay un enlace un objeto puede enviar un mensaje a otro
- Un enlace puede tener nombre, nombre de rol, navegación y agregación. Pero no multiplicidad

Enlaces y conectores (II)

- Un enlace prototípico se llama *conector*



Mensajes

- ❑ Ya sabemos que un mensaje especifica una comunicación entre objetos que transmite información con la expectativa de que se realice un servicio
- ❑ En UML se pueden modelar los siguientes tipos de mensajes:
 - ❖ Creación; Destrucción
 - ❖ Mensaje asíncrono
 - ❖ Mensaje síncrono
 - ❖ Respuesta (de un mensaje síncrono)

Diagramas de interacción

- Con los flujos de interacción se puede razonar de dos formas diferentes:
 - ❖ Centrarse en las relaciones estructurales de los objetos de la interacción y después considerar cómo se pasan los mensajes → **diagrama de comunicación**
 - ❖ Centrarse en cómo fluyen los mensajes a lo largo del tiempo → **diagrama de secuencia**

2.1. Diagrama de comunicación

□ Un diagrama de comunicación:

- ❖ destaca la organización de los objetos participantes en una interacción
- ❖ muestra el *flujo de mensajes* entre roles dentro de una comunicación



Secuenciación

- El *flujo de mensajes* forma una secuencia o *flujo de control*: un objeto envía un mensaje a otro, éste a un tercero y así sucesivamente
- Para indicar la ordenación temporal el mensaje se precede de un número. Se incrementa secuencialmente
- Se permite representar anidamiento: 1.2 es el segundo mensaje dentro del mensaje 1
- El anidamiento puede tener cualquier profundidad

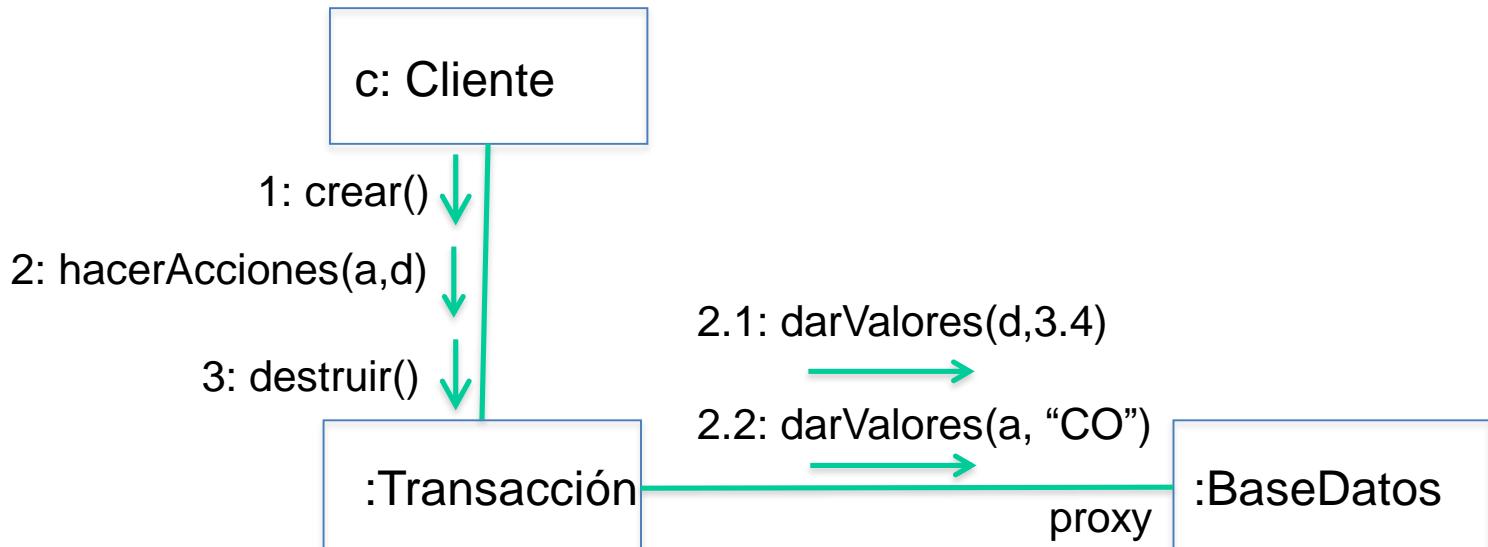
Re-formular el enlace...

- Un *enlace* representa una fuente de conocimiento de un objeto
- Un enlace se corresponde con una asociación, pero (en **Diseño**) también con:
 - ❖ Variables locales
 - ❖ Parámetros
 - ❖ Variables globales
 - ❖ Accesos propios

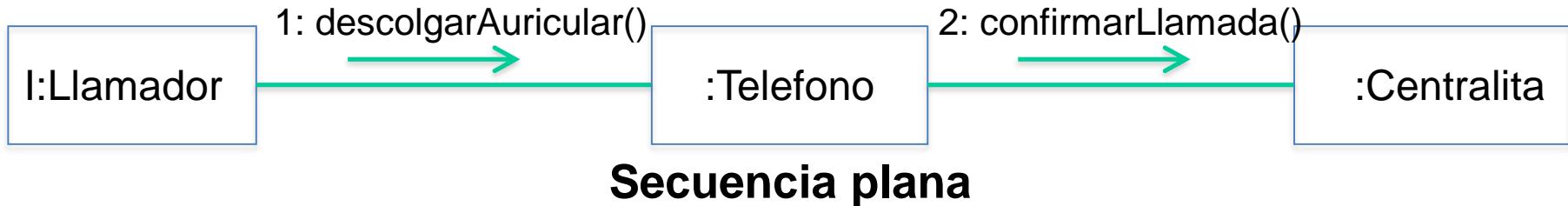
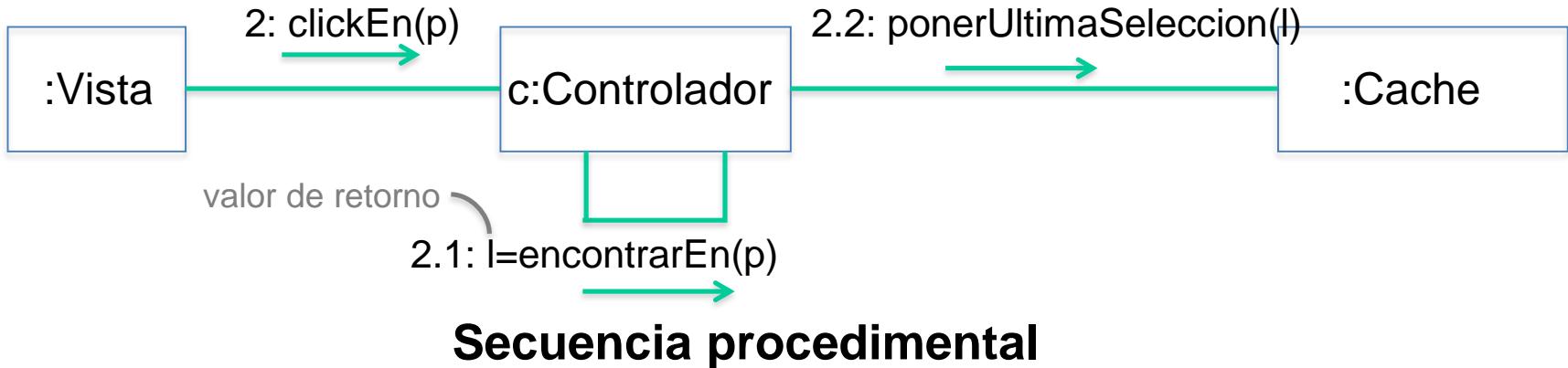
Diagrama de comunicación ...

□ Construcción del diagrama:

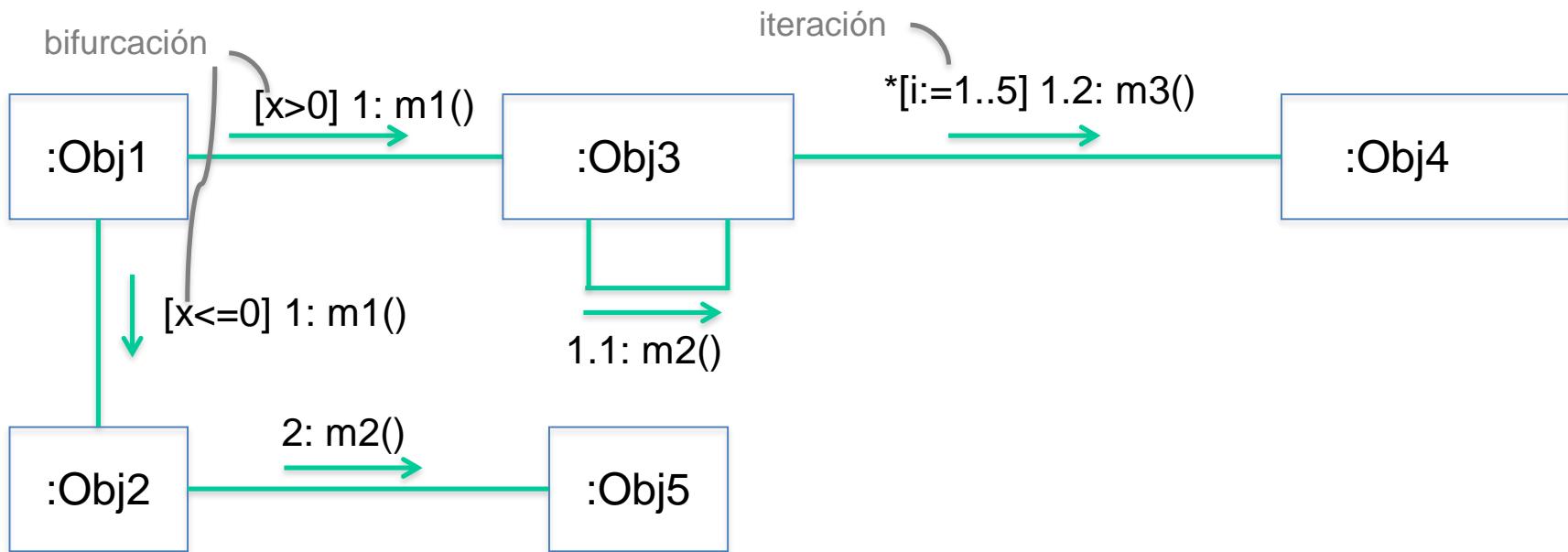
- ❖ 1. Identificamos los objetos participantes
- ❖ 2. Representamos los enlaces que los conectan
- ❖ 3. Especificamos los mensajes que los objetos envían y reciben



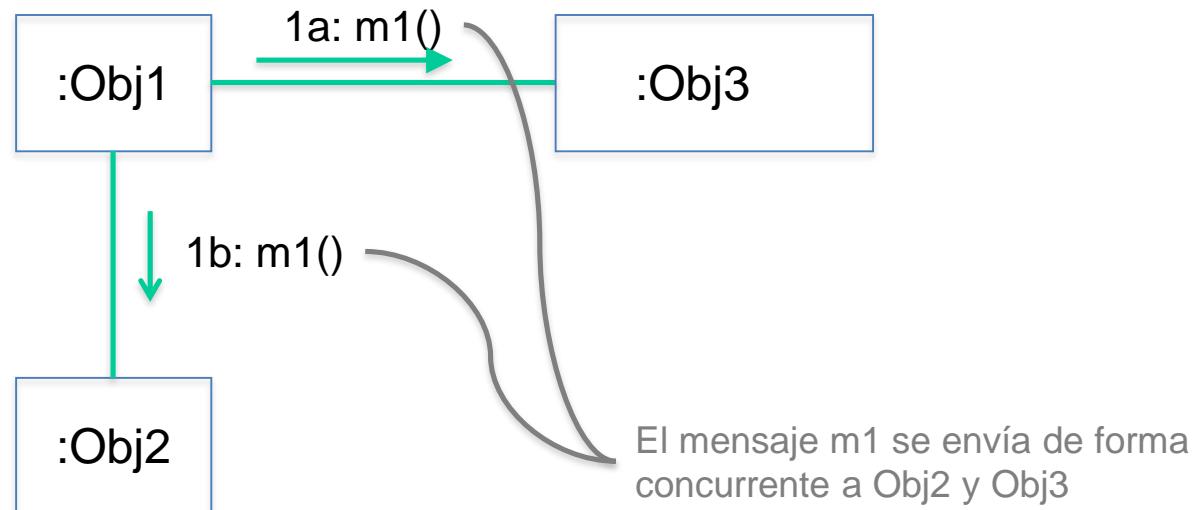
Ejemplos



Iteración y bifurcación



Concurrencia



2.2. Diagrama de secuencia

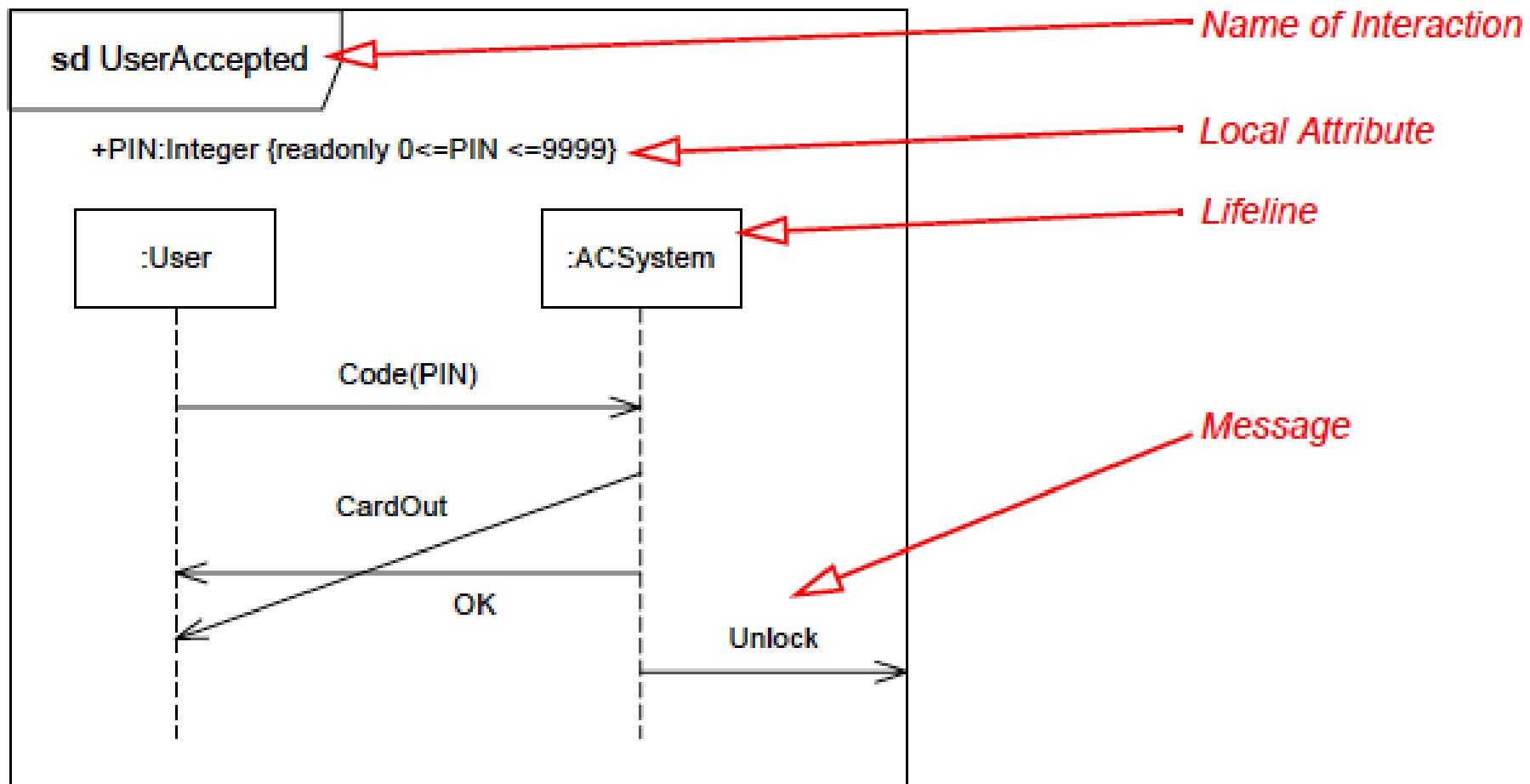
□ Utilidad

- ❖ Para la documentación de un Caso de Uso: en términos próximos al usuario y sin detallar la sincronización existente
- ❖ Para la representación precisa de las interacciones entre objetos

□ Aspectos principales de la notación gráfica

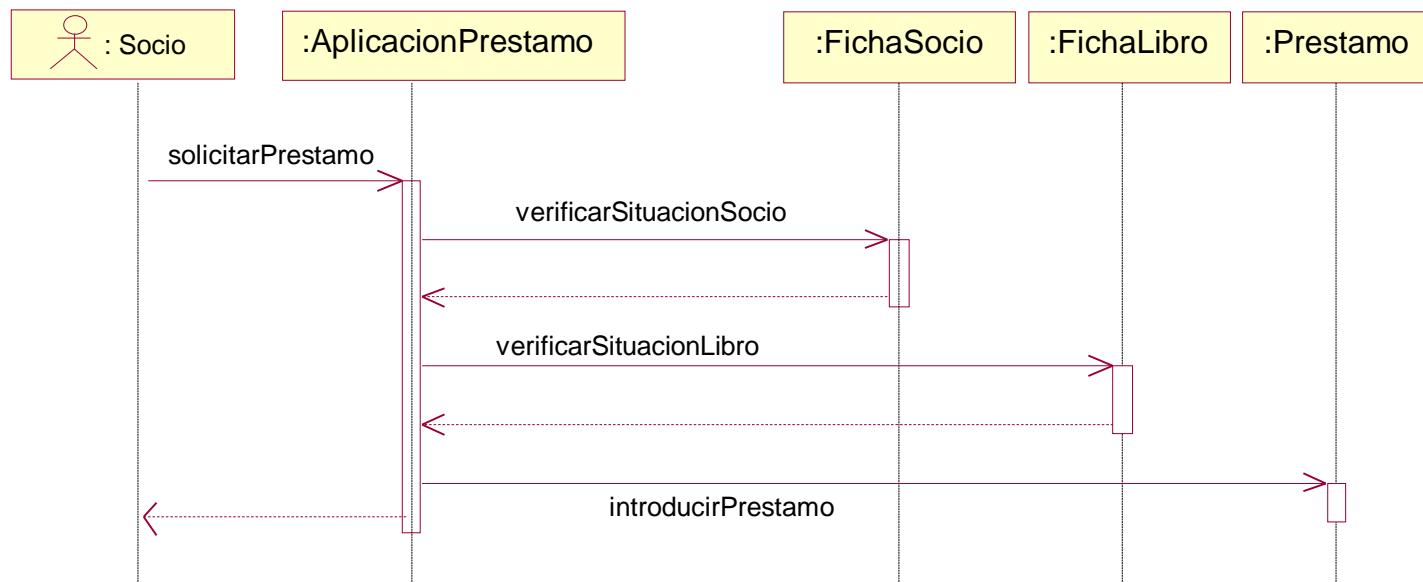
- ❖ Muestra la secuencia cronológica de mensajes entre objetos durante un escenario concreto
- ❖ Cada objeto viene dado por una barra vertical
- ❖ El tiempo transcurre de arriba abajo
- ❖ Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua

Notación



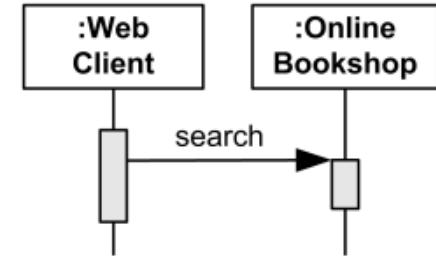
Interacción o Colaboración

- En el contexto del diagrama de secuencia:
 - ❖ Traza: secuencia de eventos.
 - ❖ *Interleaving*: dadas dos trazas, la ejecución de sus eventos podría tomar cualquier combinación. Dentro de una traza se mantiene el orden de ejecución.



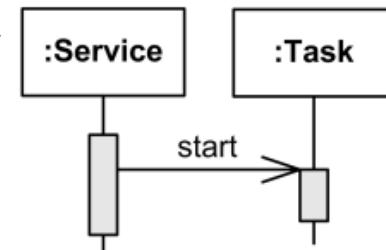
Representación de mensajes (I)

- ❑ **Mensaje síncrono:** Llamada a una operación. Se suspende la ejecución en espera de respuesta.



```
<request-message-label> ::= <message-name> '['[<input-argument-list>]']'
<input-argument-list> ::= <input-argument> [, '<input-argument>*']
<input-argument> ::= [<in-parameter-name> '='] <value-specification> / '-'
```

- ❑ **Mensaje asíncrono:** Se envía el mensaje y continúa sin esperar respuesta.

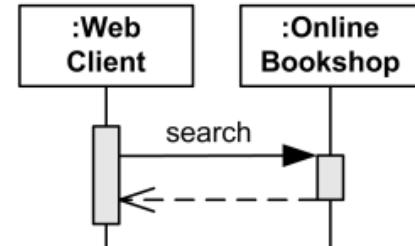


Representación de mensajes (II)

□ Respuesta a una llamada síncrona a una operación.

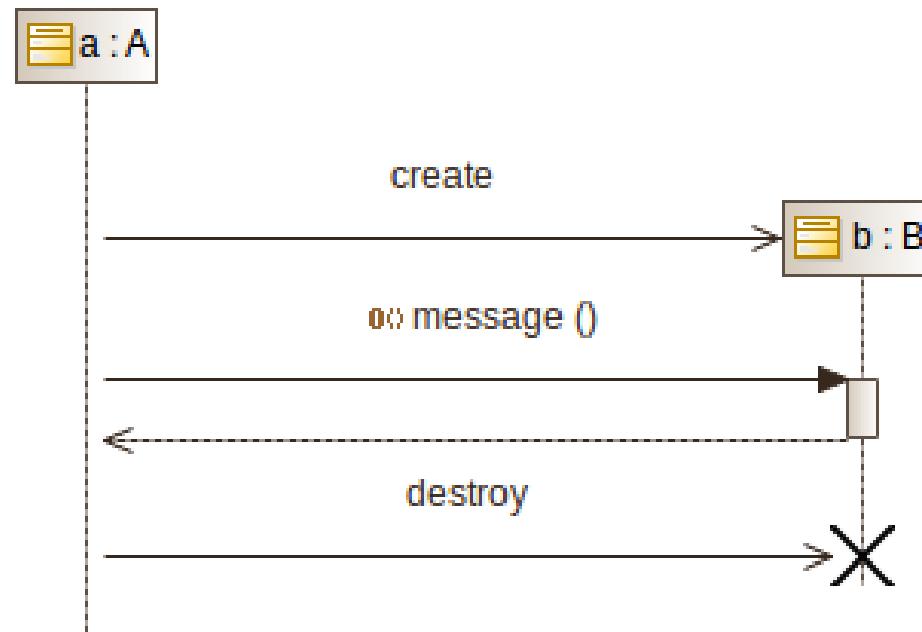
- ❖ No es obligatorio indicar el retorno del control en mensajes síncronos

```
<reply-message-label> ::= [<assignment-target> '='] <message-name>
                           [ '(' [<output-argument-list>] ')'] [':<value-specification>]
<output-argument-list> ::= <output-argument> [, <output-argument>]* 
<output-argument> ::= <out-parameter-name> ':' <value-specification> /
                           <assignment-target> '=' <out-parameter-name> [':<value-specification>]
(ver ejemplos más adelante)
```



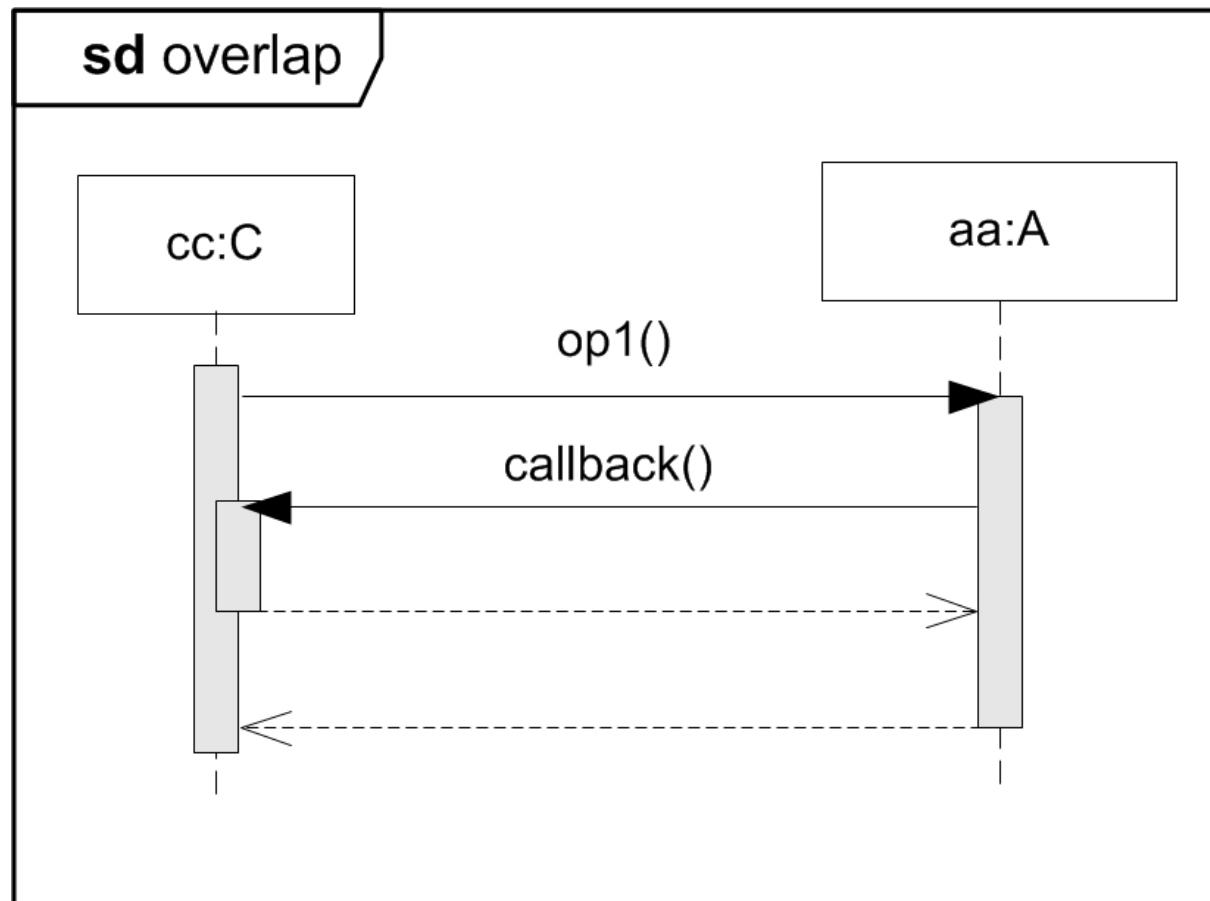
Representación de mensajes (III)

- **Mensaje de creación:** va dirigido al rectángulo del objeto y se etiqueta con *create*
- **Mensaje de destrucción:** va dirigido a la línea del objeto que se quiere destruir. El final de la flecha es una cruz y se etiqueta con *destroy*.



Representación de mensajes (IV)

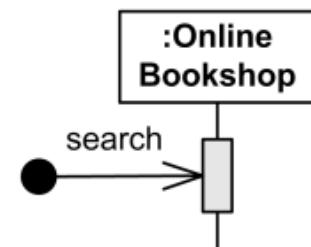
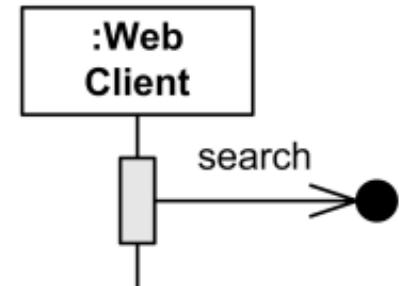
□ Solapamiento



Representación de mensajes (V)

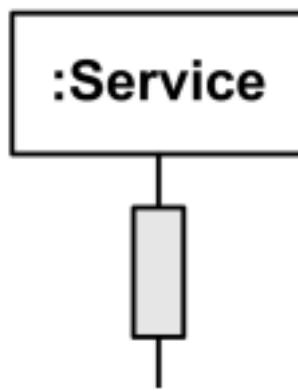
Otros tipos de mensajes:

- **Lost.** Se desconoce quien recibe el mensaje. Se interpreta como que el mensaje nunca alcanza su destinatario.
- **Found.** Se desconoce quien envía el mensaje. Se interpreta como que el emisor está fuera del ámbito que se describe.
- **Completo.** El mensaje normal.



Especificación de ejecución

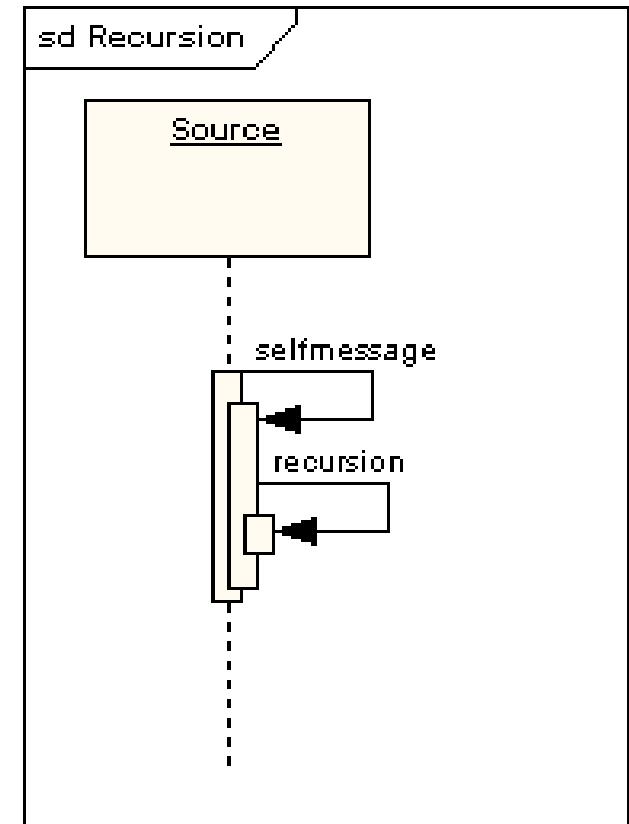
- Las bandas rectangulares representan los periodos de actividad de los objetos → Especificación de ejecución
- También se puede representar una especificación de ejecución mediante una caja rectangular más ancha y etiquetada con la acción que se ejecuta.



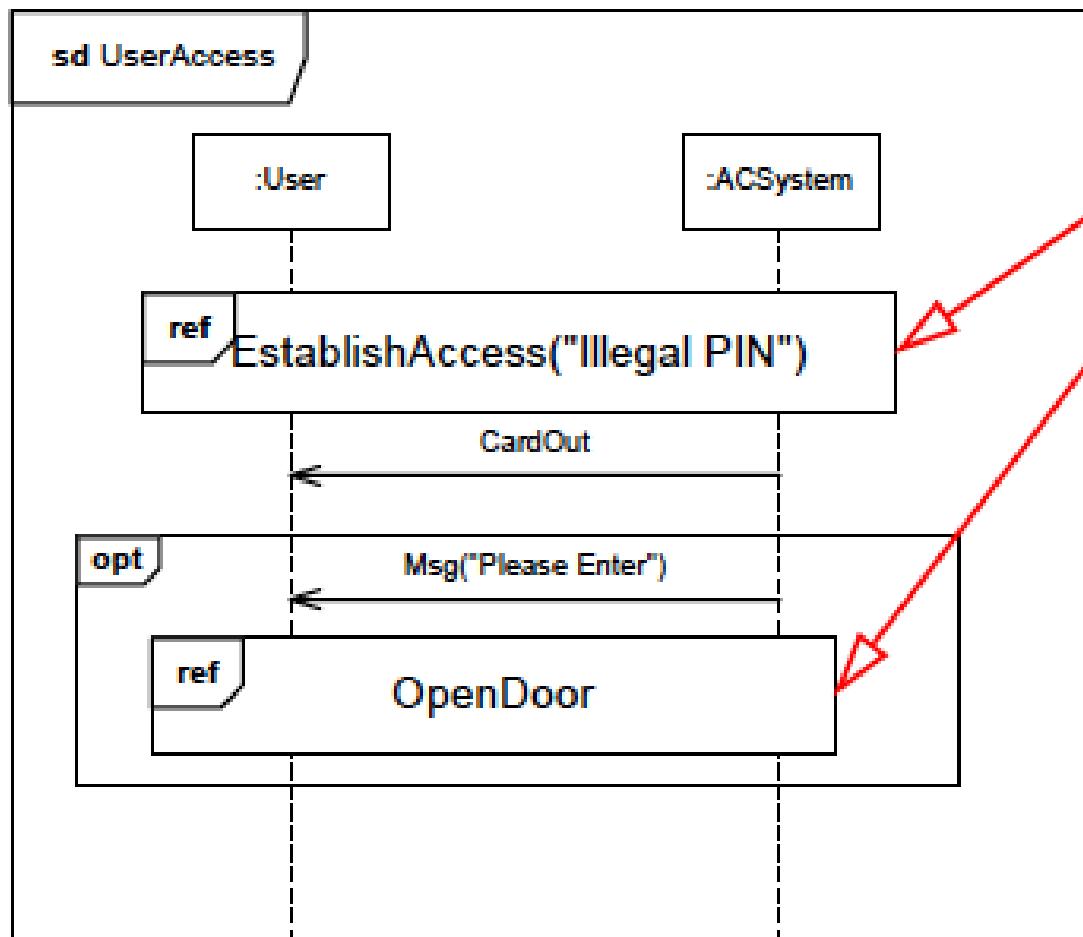
... diagrama de secuencia

- Un objeto puede enviarse a sí mismo un mensaje:

Puede representar también la entrada por parte del objeto en cierta actividad de más bajo nivel



Interacción *use o ref*

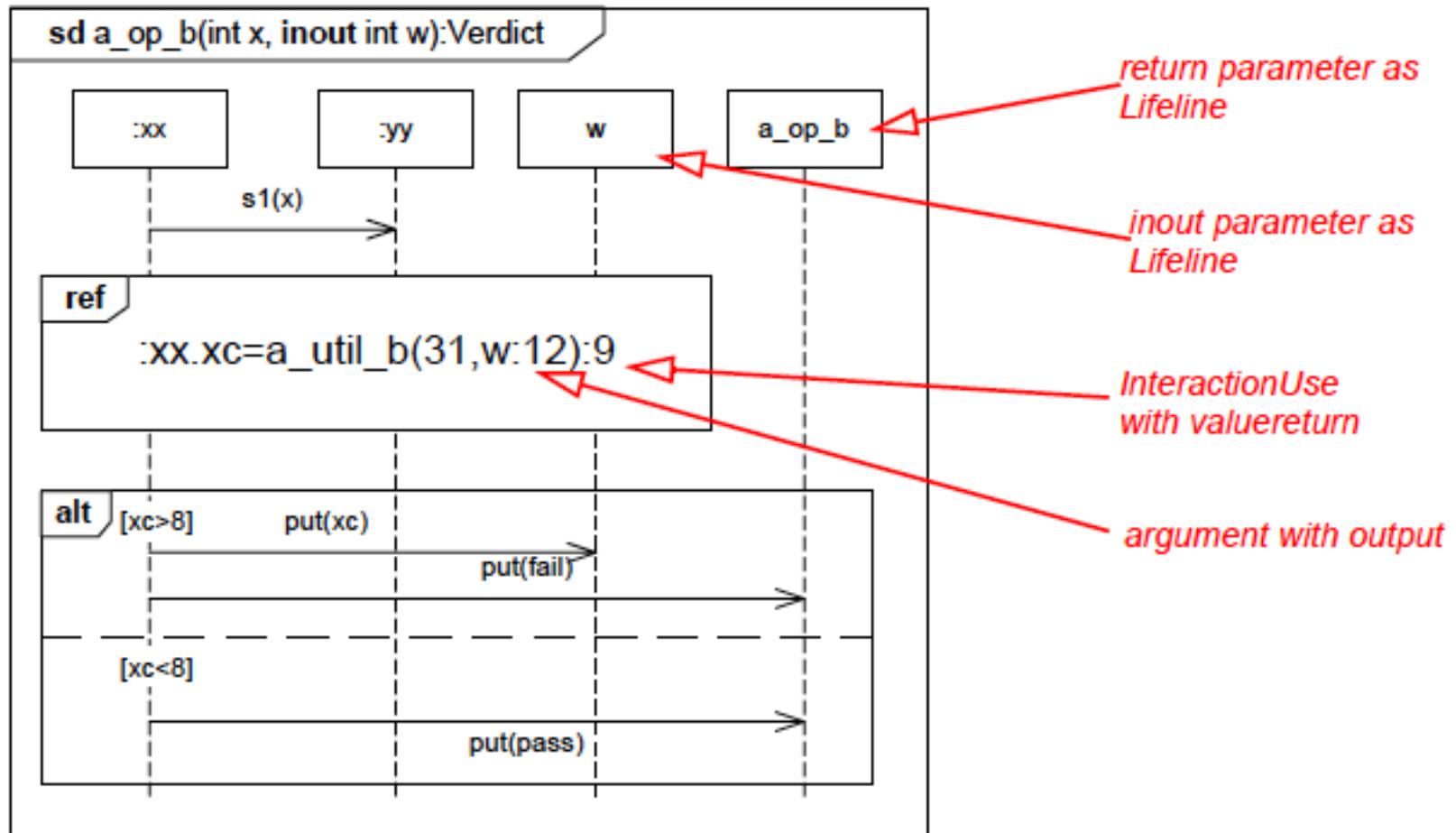


InteractionUse

Simplemente para referirnos a otro diagrama de secuencia o interacción.

Hay que sustituir los parámetros.

... parámetros

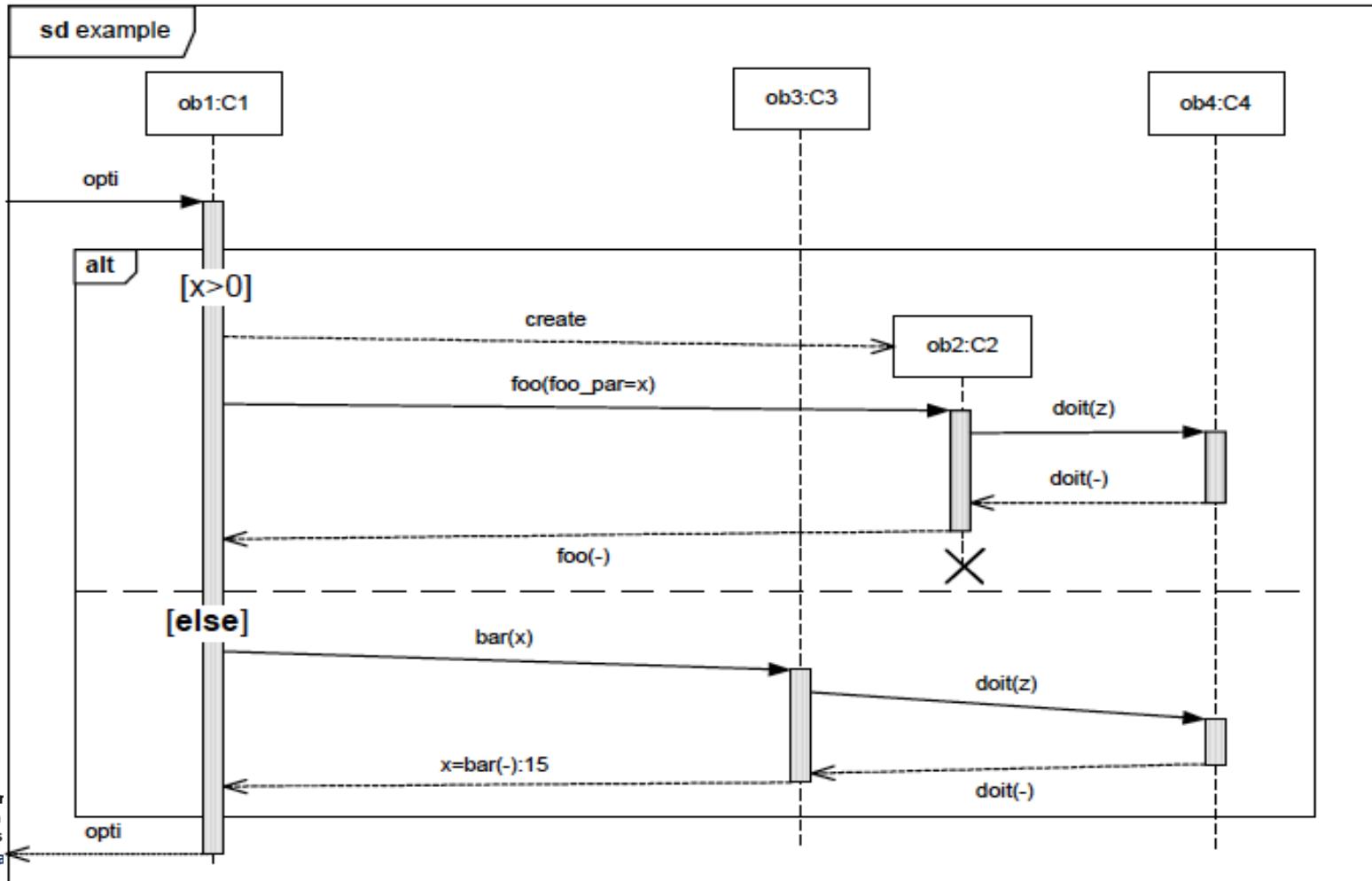


Fragmentos combinados

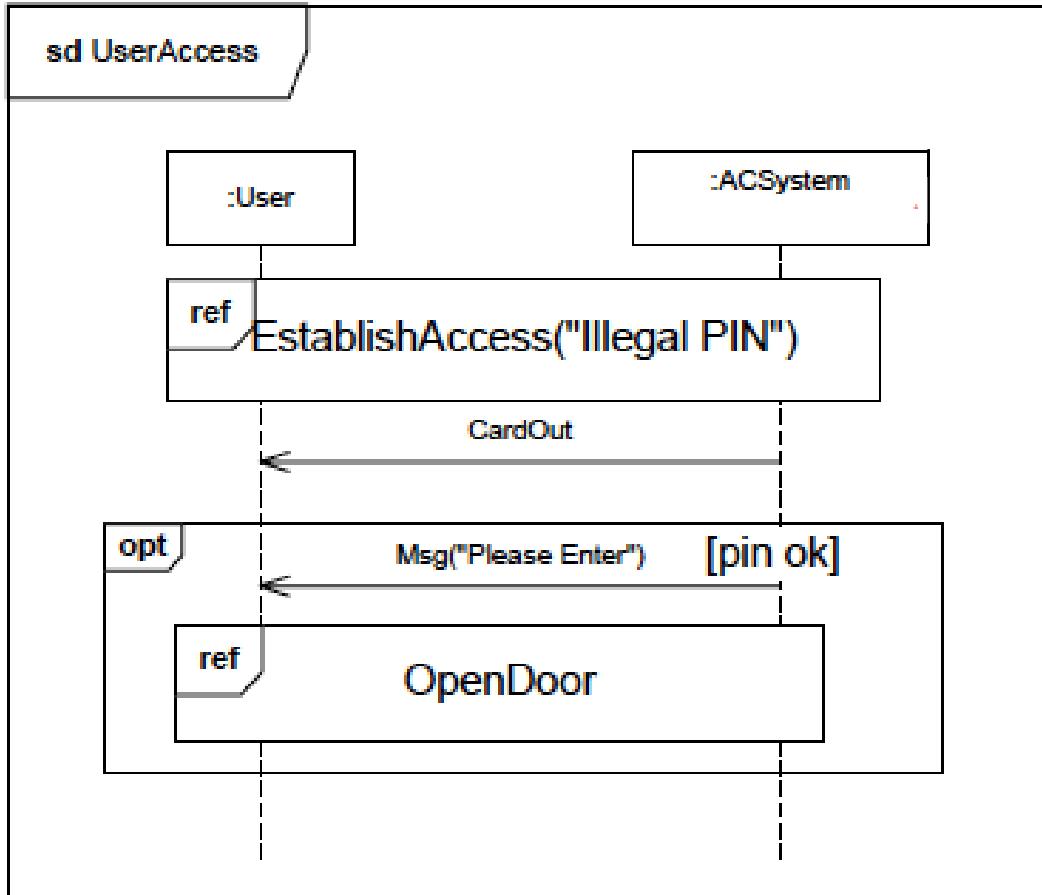
- Operadores de control.
- Sirven para describir varias trazas de manera compacta y concisa.
- Tipos:
 - ❖ alternativa (**alt**), opción (**opt**),
 - ❖ bucle (**loop**), break (**break**), continuación
 - ❖ paralelo (**par**),
 - ❖ secuencia débil (**seq**), secuencia estricta (**strict**),
 - ❖ región critica (**critical**),
 - ❖ aserción (**assert**)

Fragmento: alternativa

- Elección de comportamiento. Como máximo se ejecuta 1 operando. Guardas. Opción **else**.



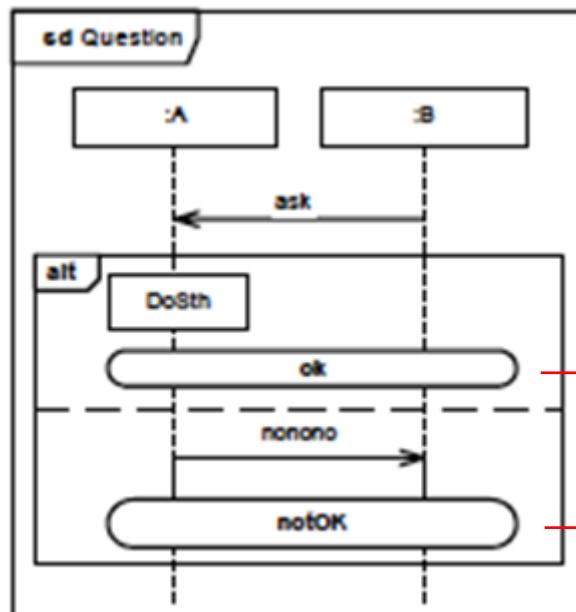
Fragmento: opción



- Representa elección de comportamiento con sólo un operando.
- Es equivalente a un **alt** con un operando no vacío y el resto vacíos.

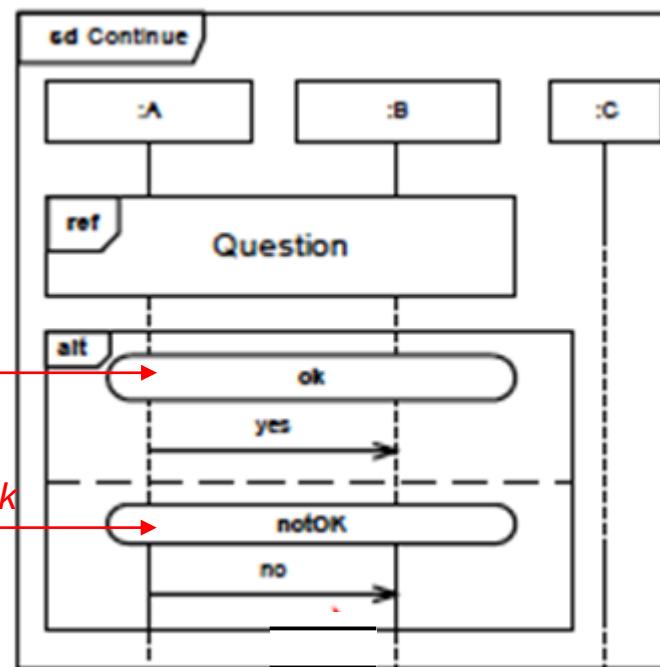
Fragmento: Continuación (I)

- ❑ Sintaxis para definir cómo continuar cuando hay diferentes ramas en un **alt**.
- ❑ *Etiquetas* que representan puntos intermedios en un flujo de control.



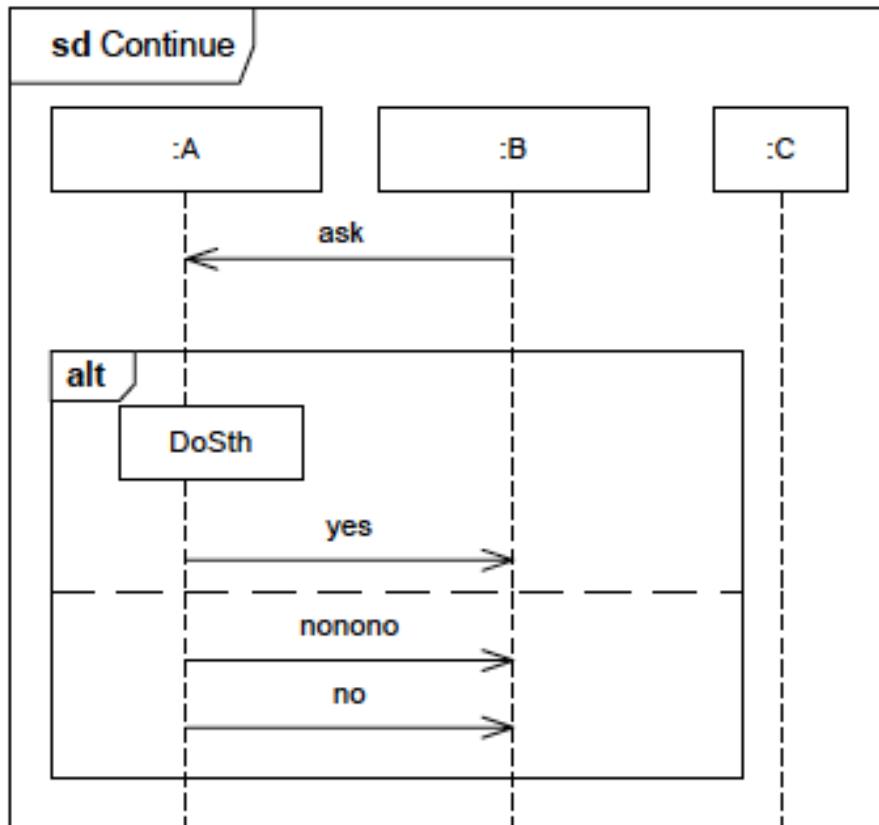
Nos falta “espacio” para indicar los mensajes después de *DoSth*, retomamos rama a partir de *ok*

Nos falta “espacio” para indicar los mensajes después de *nonono*, retomamos rama a partir de *notOK*



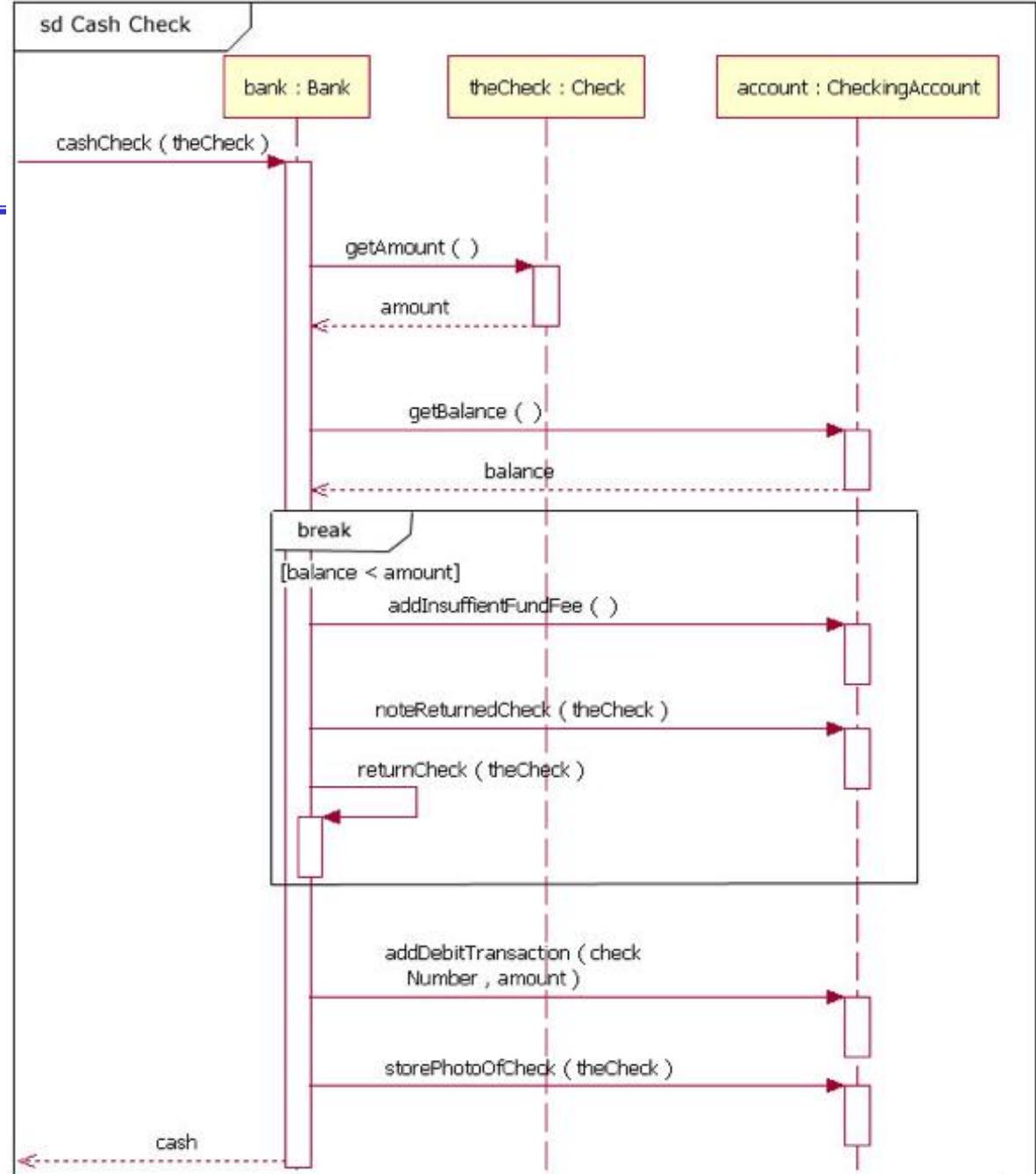
Fragmento: Continuación (II)

- Si no hubiésemos podido utilizar etiquetas de continuación, nuestro diagrama de secuencia hubiese sido similar a este



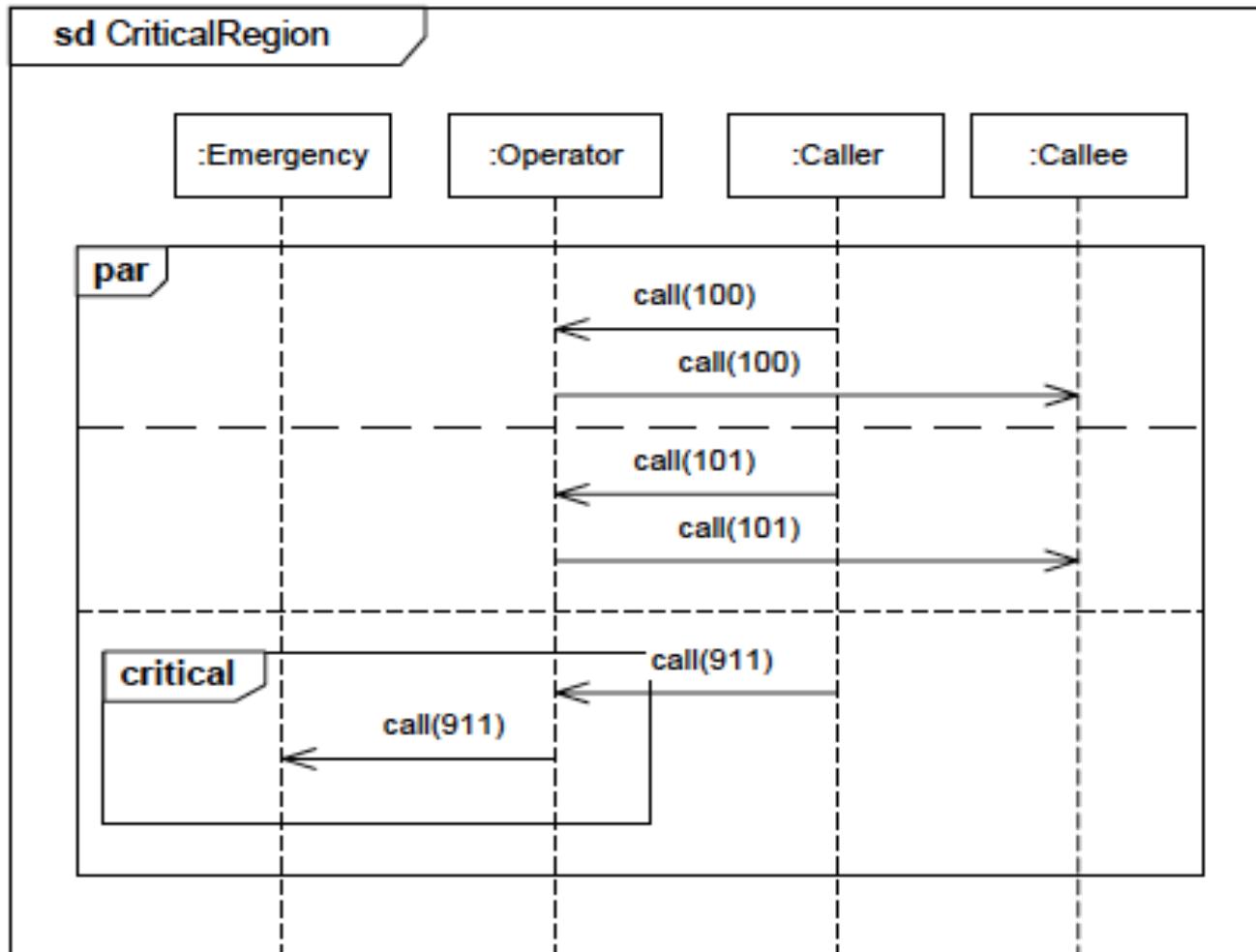
Fragmento: break

- Se ejecuta la traza de break en lugar del resto del fragmento donde está incluido.
Excepciones.



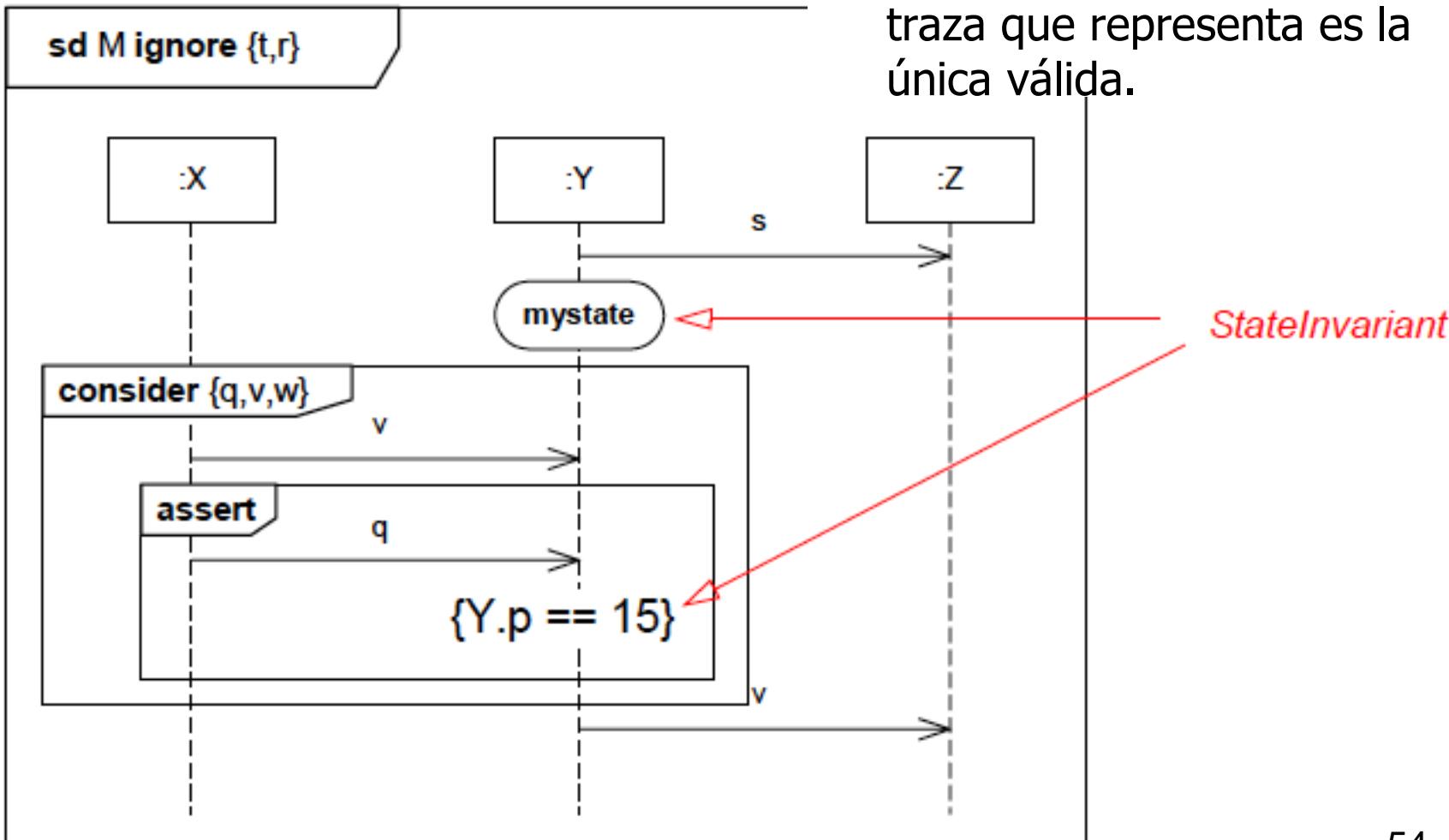
Fragmento: región crítica, paralelo

- Las trazas de la región crítica no tienen *interleaving* con el resto.
- La región es tratada de manera atómica por el fragmento que la incluye.

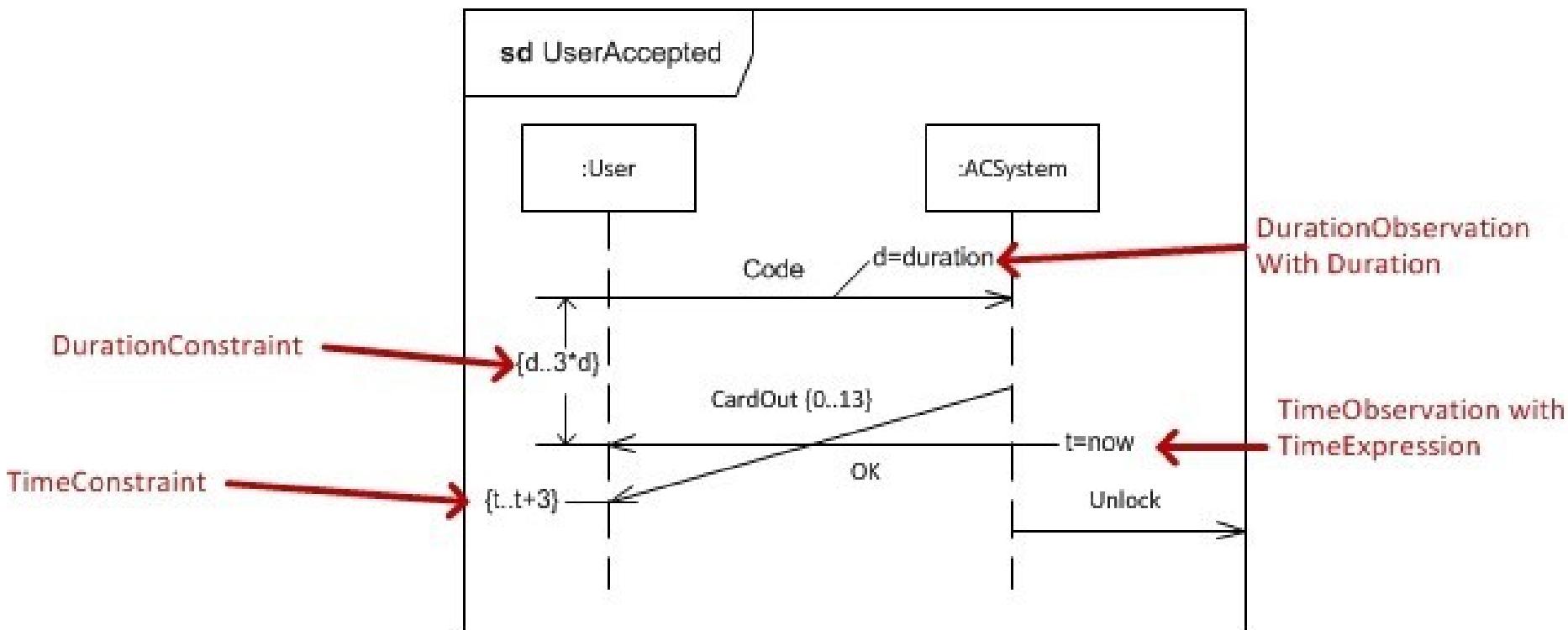


invariantes

- **ignore**. Lista de mensajes que se pueden obviar.
- **consider**. Lista de mensajes a considerar. Los otros se pueden obviar.
- **assert** = aseveración. La traza que representa es la única válida.

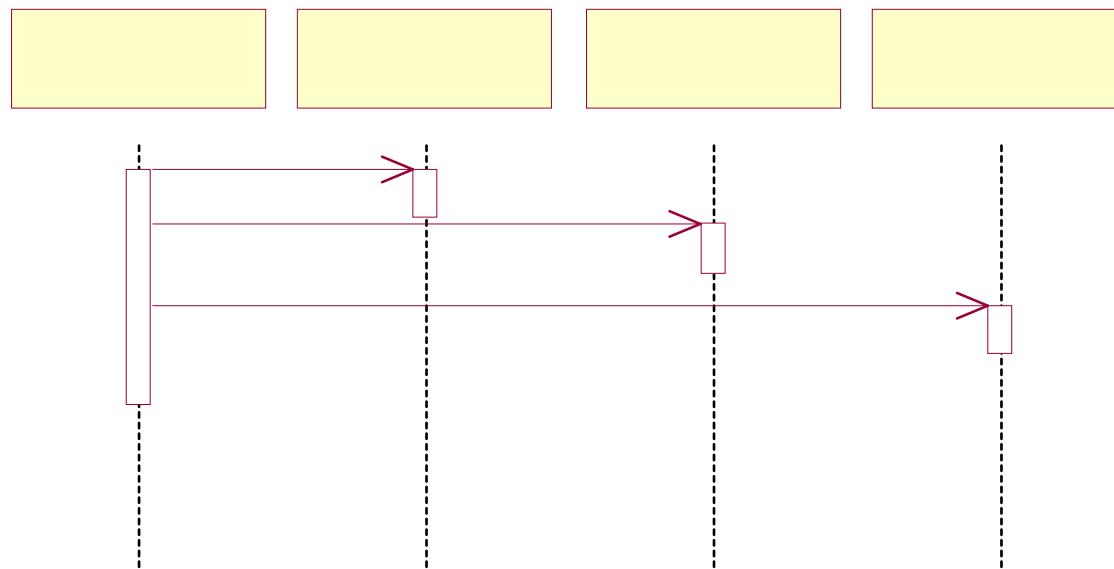


tiempo



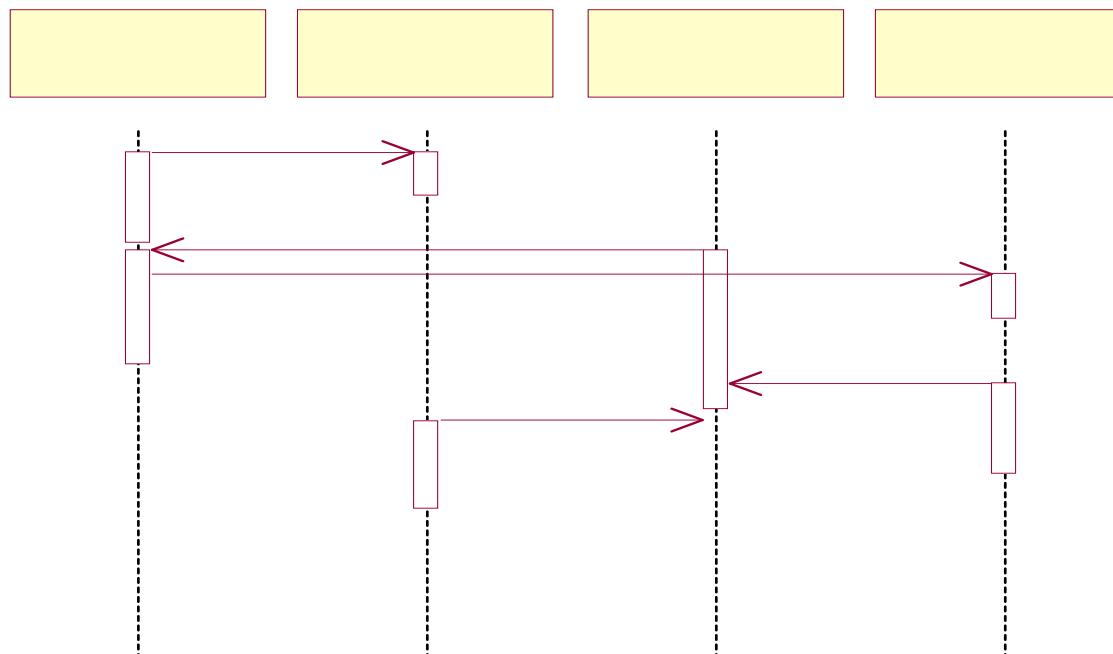
Tipos de control (I)

- El Diagrama de Secuencia refleja de manera indirecta las opciones de control
- Un control centralizado tiene una forma como esta (estructura tipo *tenedor*):



Tipos de control (II)

- Un control descentralizado tiene una forma como esta (estructura tipo *escalera*):



3. Diagramas de Estados

- ❑ Con los diagramas de estados podemos ver el sistema como un conjunto de objetos autónomos y concurrentes ...
- ❑ Los diagramas de estados modelan el comportamiento *interno* de los objetos
- ❑ Todos los objetos de una clase tienen el mismo patrón de comportamiento → Ese patrón es modelado por un diagrama de estado
 - ❖ Por tanto, un diagrama de estados modela el comportamiento de una única clase → Todos los objetos comparten el mismo diagrama de estados

Diagrama de Estados ...

- Creamos diagramas de estados sólo para las clases con **comportamiento dinámico relevante**
- Para el resto de objetos podemos suponer que sólo poseen **un** estado

Diagrama de Estados ...

- Los diagramas de estados de UML:
 - ❖ Utilizan el formalismo de los *statechart* de Harel
 - ❖ Son autómatas de estados finitos adaptados al modelo orientado a objetos
- Modelamos objetos que responden a eventos discretos → tienen por tanto comportamiento reactivo

Estados y Transiciones

- Un diagrama de estados relaciona los *estados* en los que puede estar un objeto con los *eventos* que pueden sucederle
- Un cambio de estado causado por un evento se llama *transición*

Ejemplo de D.E. para un objeto alarma:

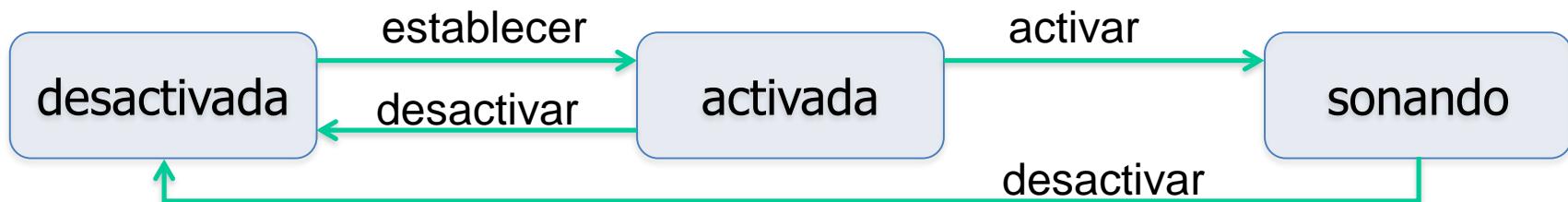


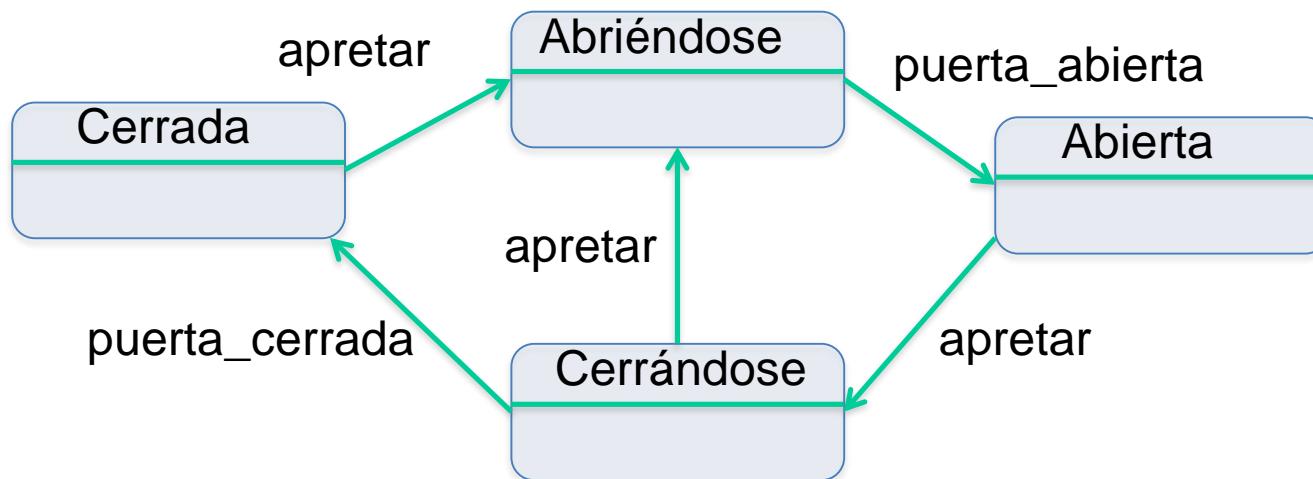
Diagrama de Estados ...

- Un diagrama de estados es un grafo, los nodos representan los estados y los arcos dirigidos son transiciones rotuladas con nombres de eventos
- Si un objeto está en un estado y le llega un evento que rotula una de sus transiciones entonces la transición se *dispara* y el objeto *cambia de estado*
- El diagrama de estados define un modelo *reactivo*

Estados

- ☐ Cuando el objeto recibe un evento, el estado siguiente depende del actual así como del evento

Ejemplo de D.E. para un objeto de tipo Puerta Garaje (el D.E. describe el control del motor de la puerta):



Estados ...

- ❑ Si se produce un evento que no rotula ninguna transición que parte del estado actual entonces el evento *se pierde*
- ❑ El diagrama de estados especifica la secuencia de estados por los que pasa un objeto como consecuencia de una secuencia de eventos

Estados ...

- ❑ Los estados no se definen sobre todos los atributos del objeto
- ❑ Cada objeto posee su propio estado que es el resultado de los eventos que ha recibido
- ❑ Cada objeto es independiente y *evoluciona* a su manera

Diagrama de Clases:

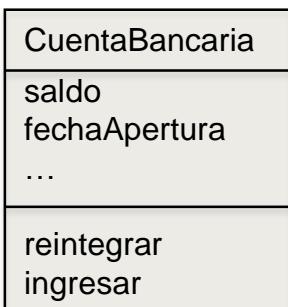
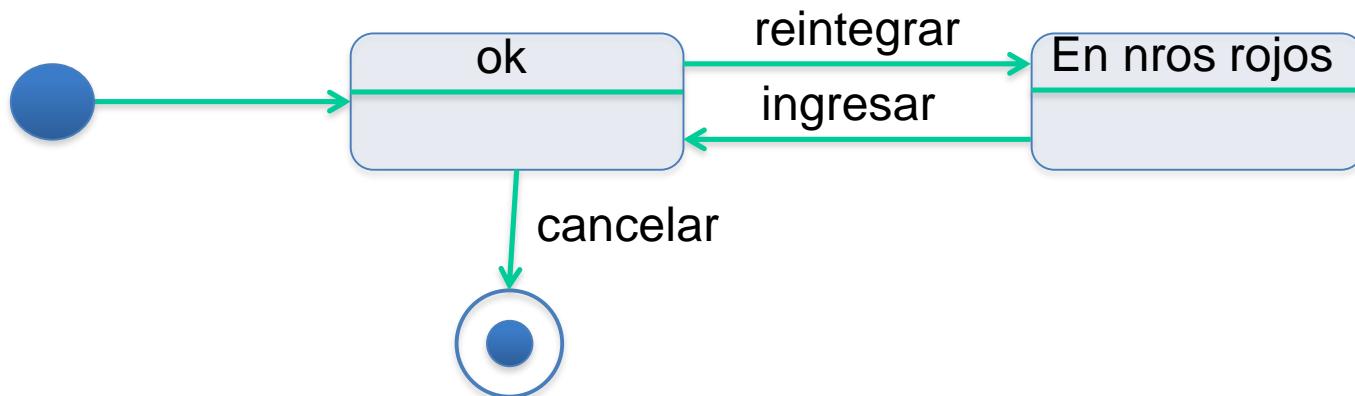


Diagrama de Estados:

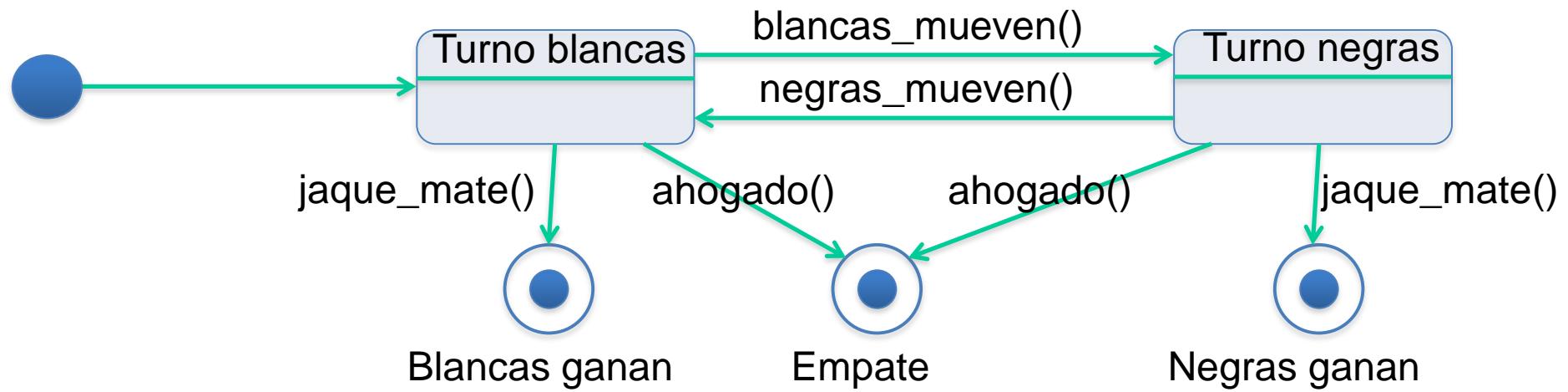


Estados ...

- ❑ Los diagramas de estados pueden representar objetos con *vidas* finitas, desde que el objeto nace hasta que muere
- ❑ Pueden tener un pseudo-estado inicial y estados finales

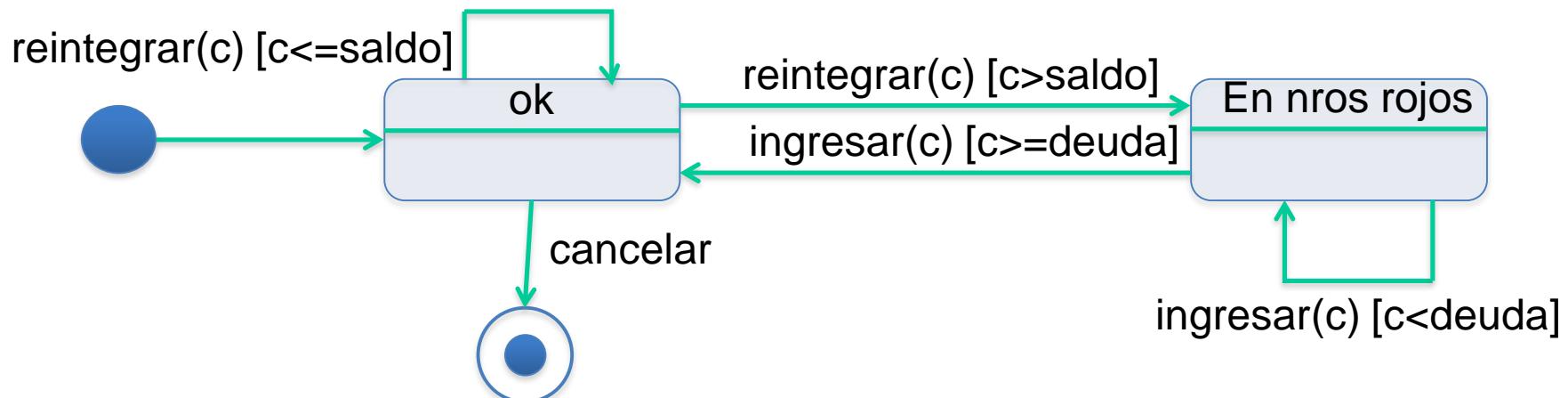


Ejemplo (juego del ajedrez simplificado)

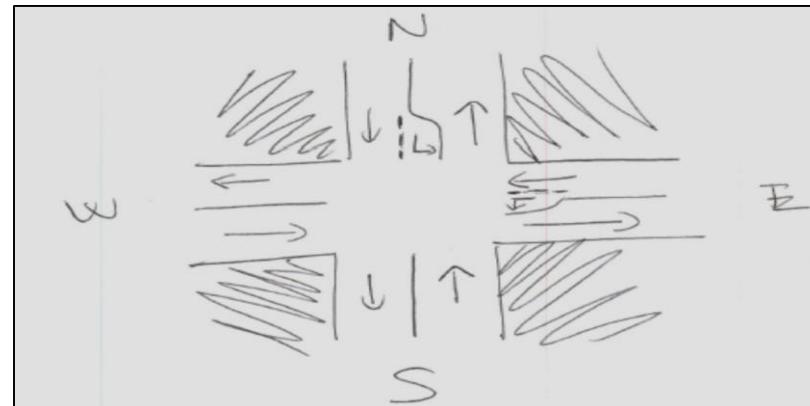
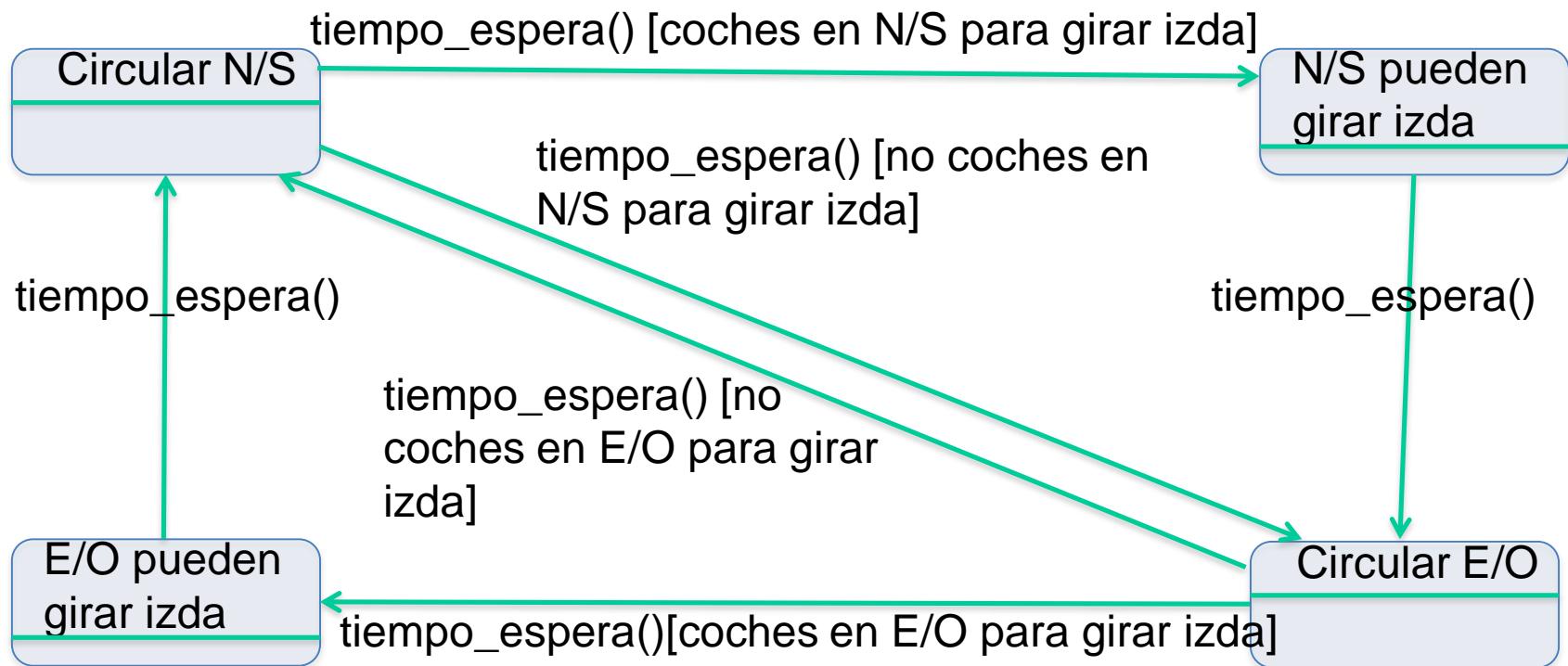


Condiciones /guardas

- ❑ Una condición es una función lógica que normalmente construiremos con los valores de los atributos del objeto y los parámetros de los eventos
- ❑ Los eventos no tienen duración. Sin embargo las condiciones son válidas durante un intervalo de tiempo
- ❑ Estar en un estado es una condición



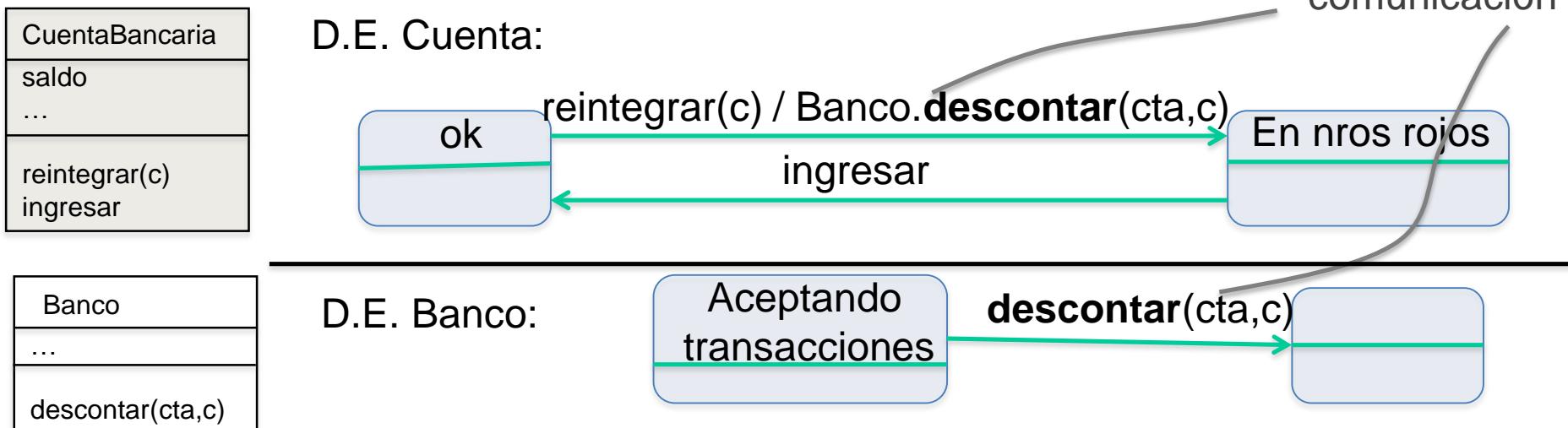
Ejemplo (control de semáforos en una intersección de 2 vías)



Envío de eventos

- ¿De dónde llegan los eventos a los objetos?
 - ❖ De fuera del sistema, los envían los **actores** → mensajes
 - ❖ De **otros objetos** del propio sistema → mensajes

- Los objetos se envían eventos entre sí para solicitar peticiones (servicios)



Envío de eventos ...

- ❑ El sistema lo vemos como una sociedad de objetos que evoluciona intercambiando eventos (enviándose mensajes)
- ❑ El modelo dinámico lo podemos ver como una colección de diagramas de estados que interactúan entre sí (se comunican) a través de eventos
- ❑ El envío de eventos introduce un modelo *proactivo*

Operaciones

- ❑ Los diagramas de estados además de mostrar secuencias de eventos especifican qué hace el objeto como respuesta a esos eventos → Operaciones
- ❑ Las operaciones realizan computaciones
- ❑ Podemos modelar dos tipos de operaciones:
 - ❖ Acciones
 - ❖ Actividades

Acciones

- Las acciones son las operaciones que consideramos instantáneas o inmediatas → su duración es insignificante en comparación con la resolución del diagrama (con la vida del objeto)
- Suponen una computación o el envío de un evento
- Por supuesto hasta la operación más sencilla requiere tiempo. Las consideramos instantáneas con respecto a la granularidad de los eventos reales

Acciones ...

- Van asociadas a:
 - ❖ Las transiciones: ***evento [guarda] / acción***
 - Evento: invocación a una operación de la clase (mensaje de entrada a la línea de vida del objeto en un diagrama de secuencia)
 - Se pueden indicar varios eventos separados por comas
 - Guarda: expresión booleana en base a los parámetros del evento y el contexto del objeto (atributos y enlaces)
 - Acción: invocación a una operación de la propia clase o de otra clase (mensaje de salida desde la línea de vida del objeto en un diagrama de secuencia)
 - Cuando invocamos a operaciones de otra clase necesitamos indicarlo explícitamente
 - ✓ *<Nombre de otra clase>.operación*
 - ✓ *<referencia a objeto de otra clase>.operación*



...

Acciones ...

❖ ...

❖ Los estados:

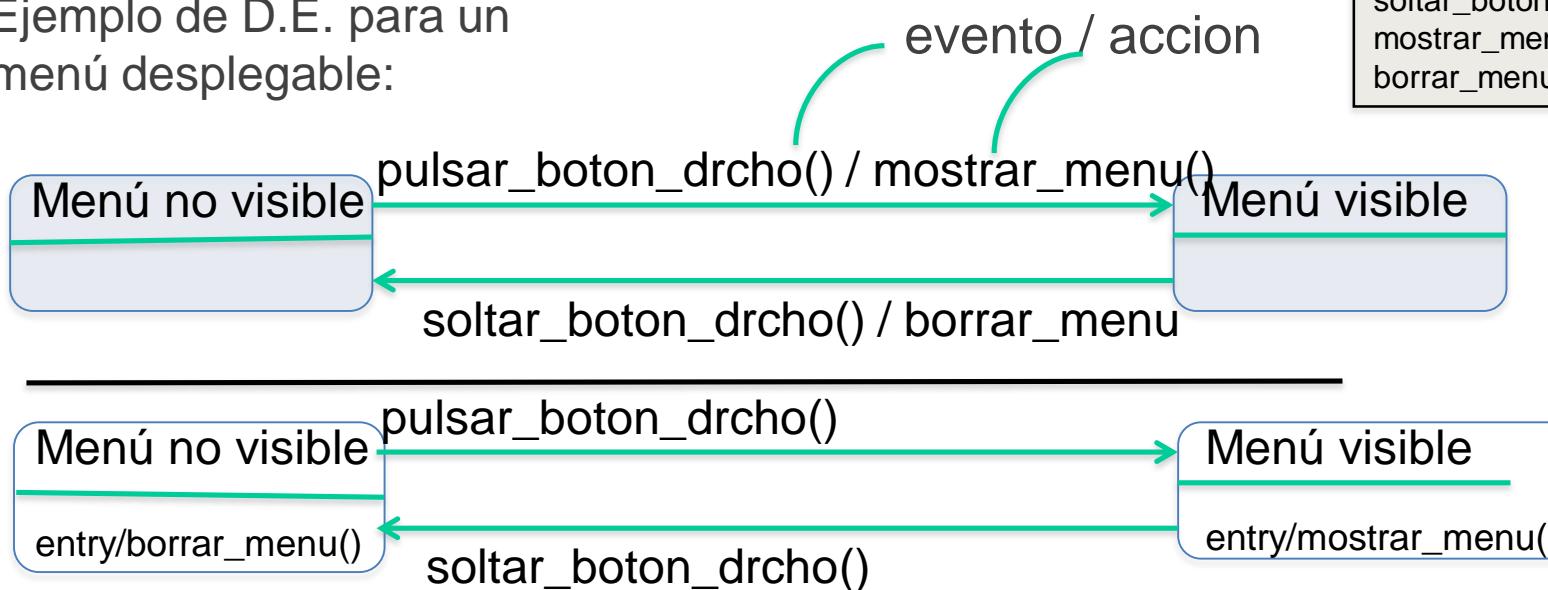
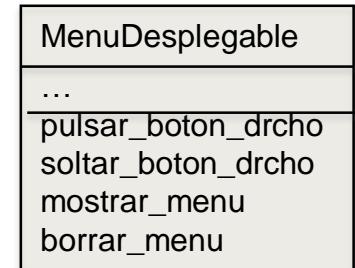
➤ Acción de entrada:

○ **entry / acción**

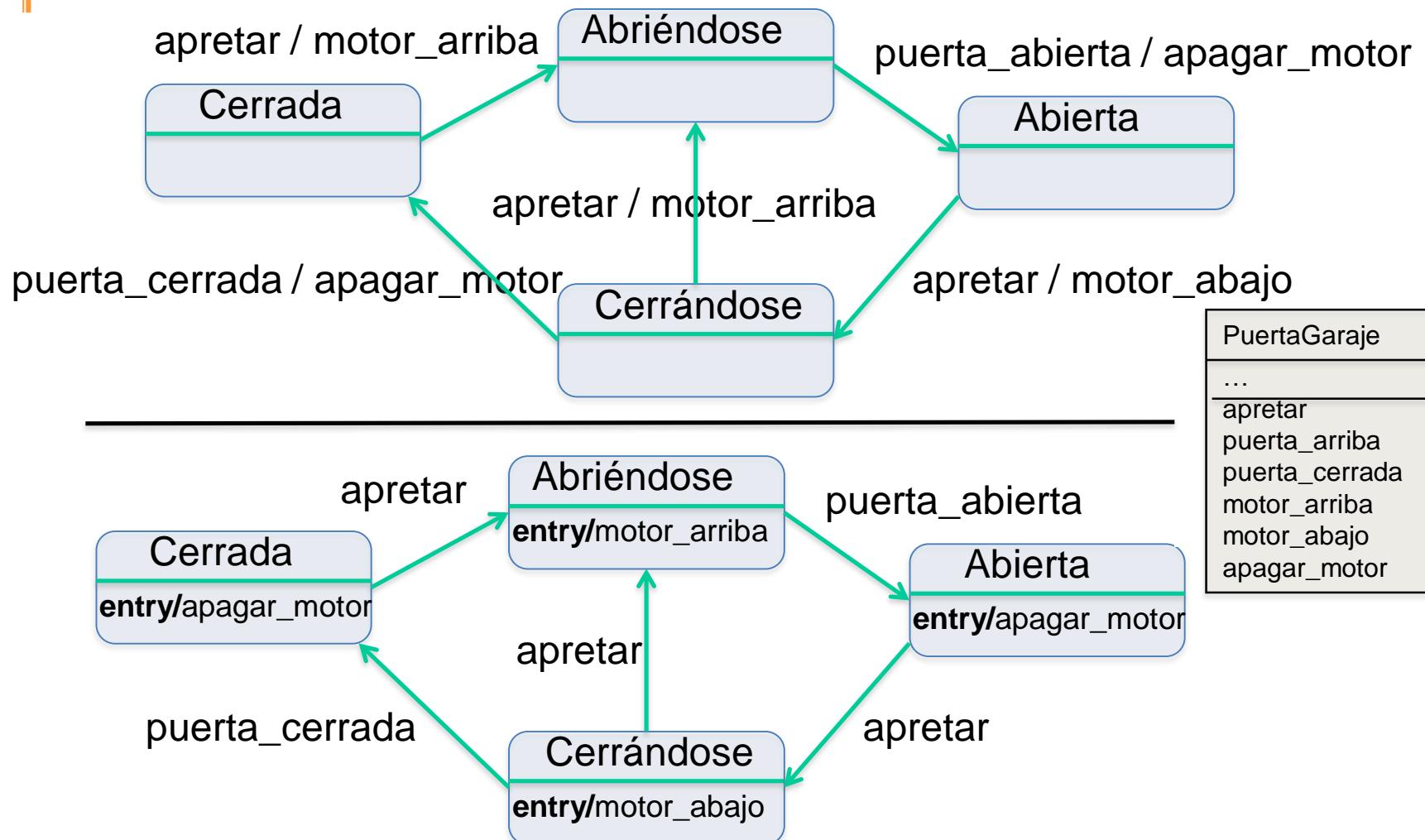
➤ Acción de salida:

○ **exit / acción**

Ejemplo de D.E. para un menú desplegable:



Acciones ...

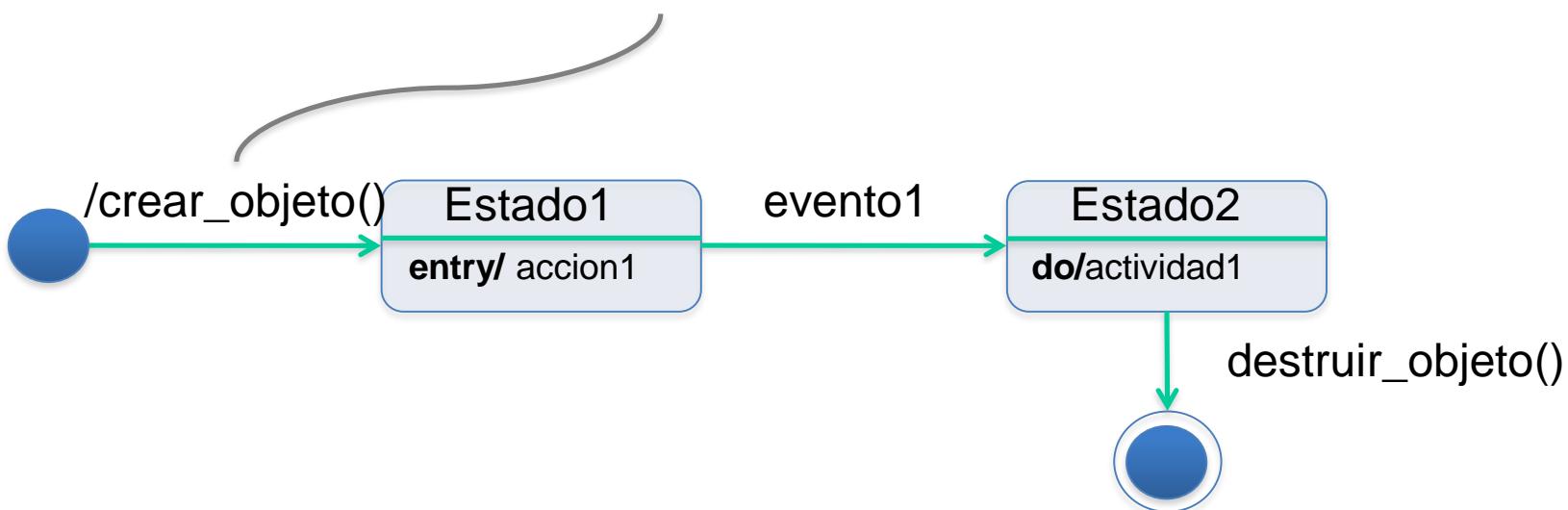


Actividades

- Son operaciones cuya realización requiere un cierto tiempo
- Están asociadas a los estados
- Ejemplos: mostrar una imagen, llevar a cabo un cálculo
- La actividad comienza en cuanto el objeto entra en el estado
- La actividad o bien termina por sí misma o bien es abortada por un evento antes de su finalización

Actividades ...

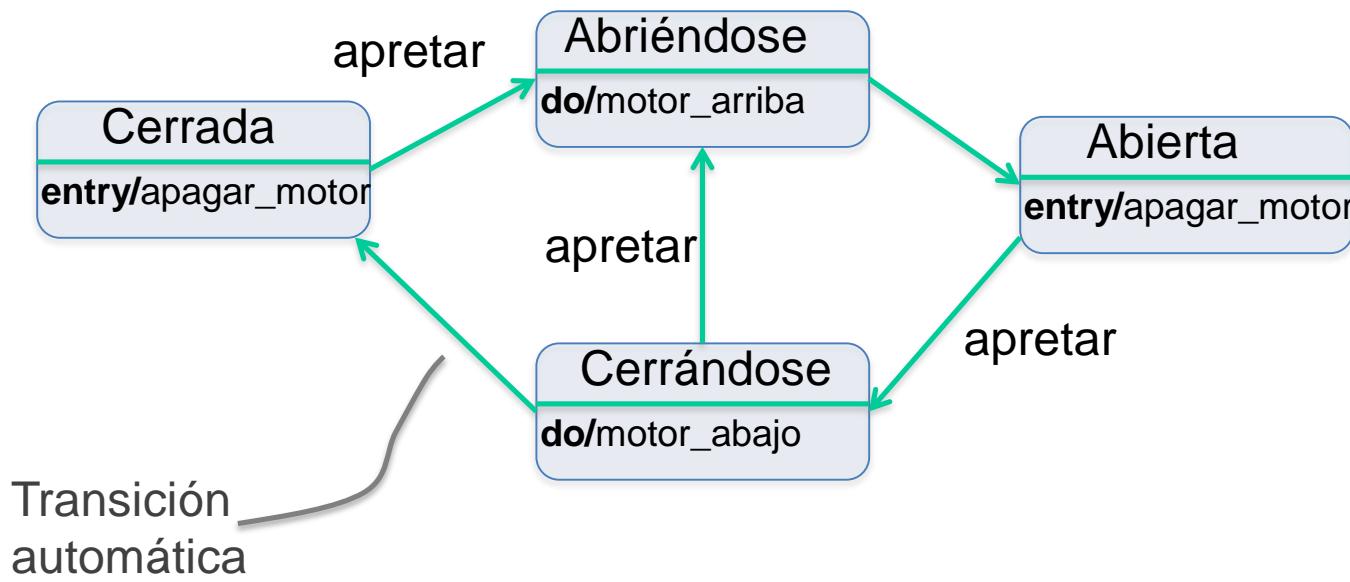
La transición asociada a un pseudo-estado inicial no puede tener un evento, pero sí una acción



Transiciones automáticas

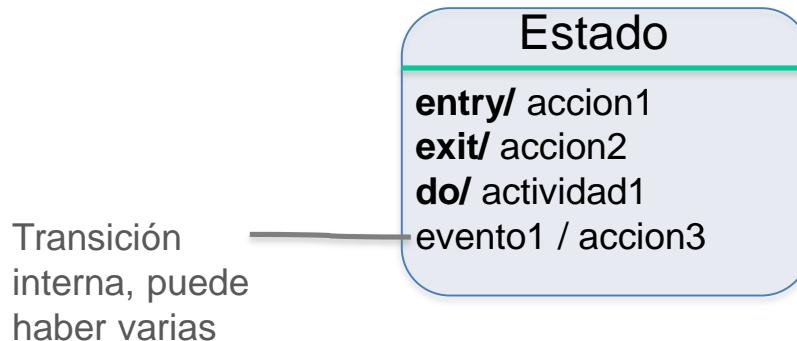
- Una transición sin nombre de evento es una transición automática
- Se dispara cuando las acciones y la actividad asociada al estado se completan

Transiciones automáticas ...



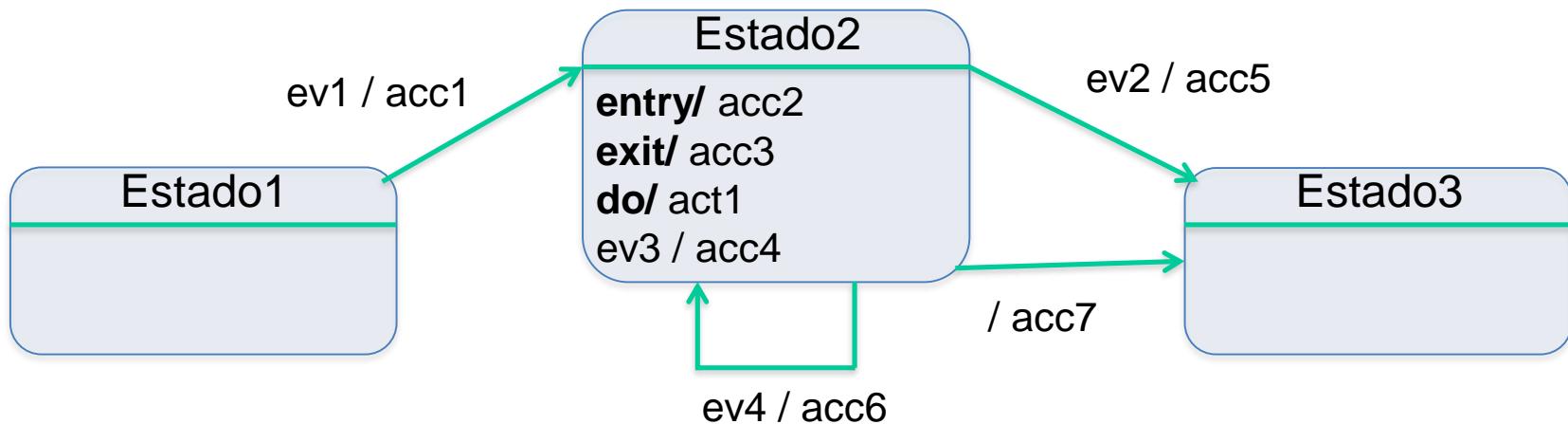
Transiciones internas

- ❑ Podemos modelar transiciones que no suponen un cambio de estado
- ❑ El evento que las etiqueta es aceptado, si tienen acción asociada se ejecuta
- ❑ Son diferentes a las auto-transiciones



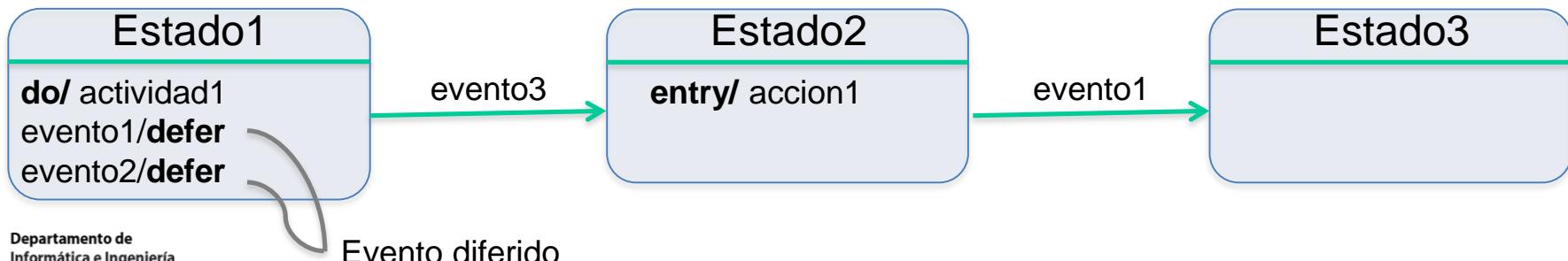
Modelo de ejecución

- ❑ ¿Cuál es la secuencia de acciones que se ejecutan al dispararse *ev1* en *Estado1*?
- ❑ ¿Qué eventos pueden ocurrir estando ejecutando *act1* en *Estado2*? ¿Qué acciones se desencadenan?



Eventos diferidos

- ❑ En ocasiones algún evento no puede ser gestionado en el estado actual pero tampoco desecharlo
- ❑ Ese evento hay que gestionarlo en otro estado
- ❑ Los eventos diferidos se “guardan” hasta que se llega a un estado con una transición que lo puede utilizar



Estructuración de diagramas

- ❑ Los diagramas de estados se estructuran para hacer posible descripciones concisas de sistemas complejos
- ❑ Expandir las actividades y/o las acciones utilizando diagramas de actividad

Estructuración de diagramas ...

□ Generalización de estados

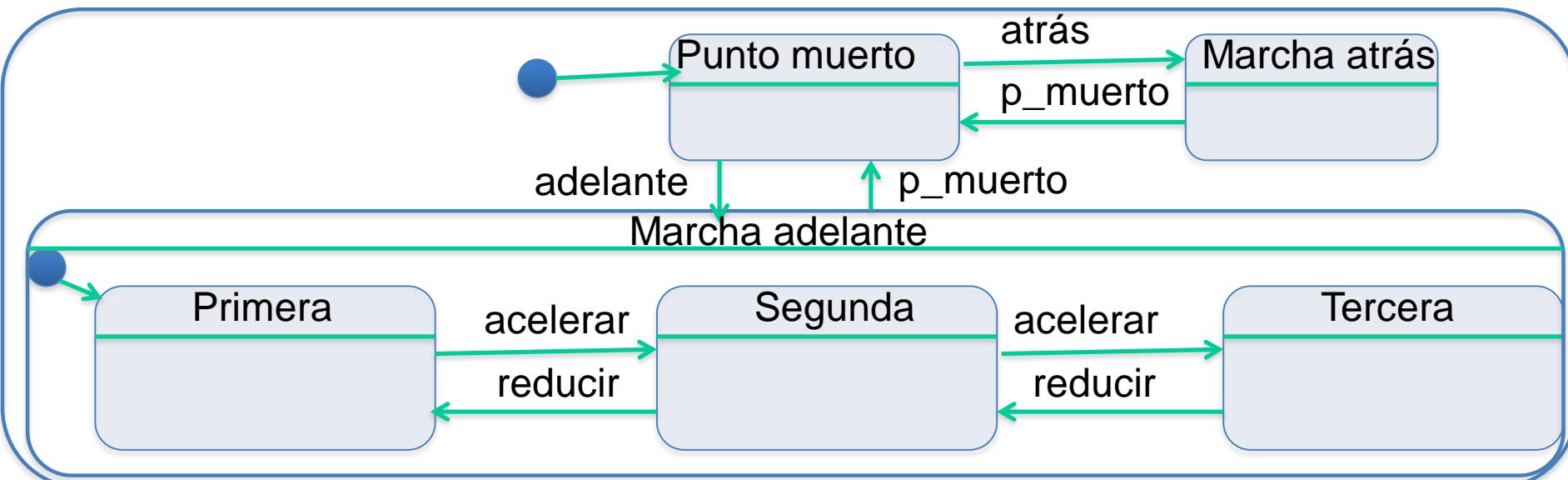
- ❖ Los estados pueden poseer sub-estados que heredan las transiciones de sus super-estados

Ejemplo de D.E. para el control del cambio de marchas automático en un coche:

R - Rear (marcha atrás)

N - Neutral (punto muerto)

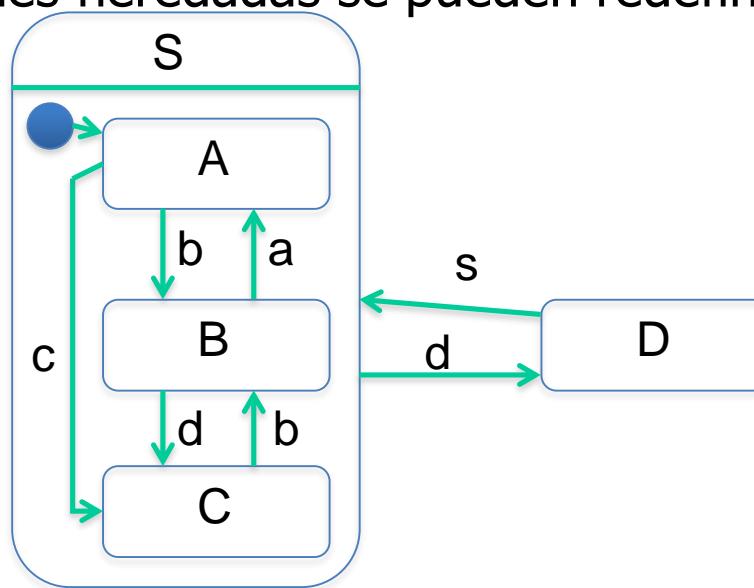
D - Drive (marcha adelante): 1,2,3



Estructuración de diagramas ...

□ Generalización ...

- ❖ Las transiciones heredadas se pueden redefinir

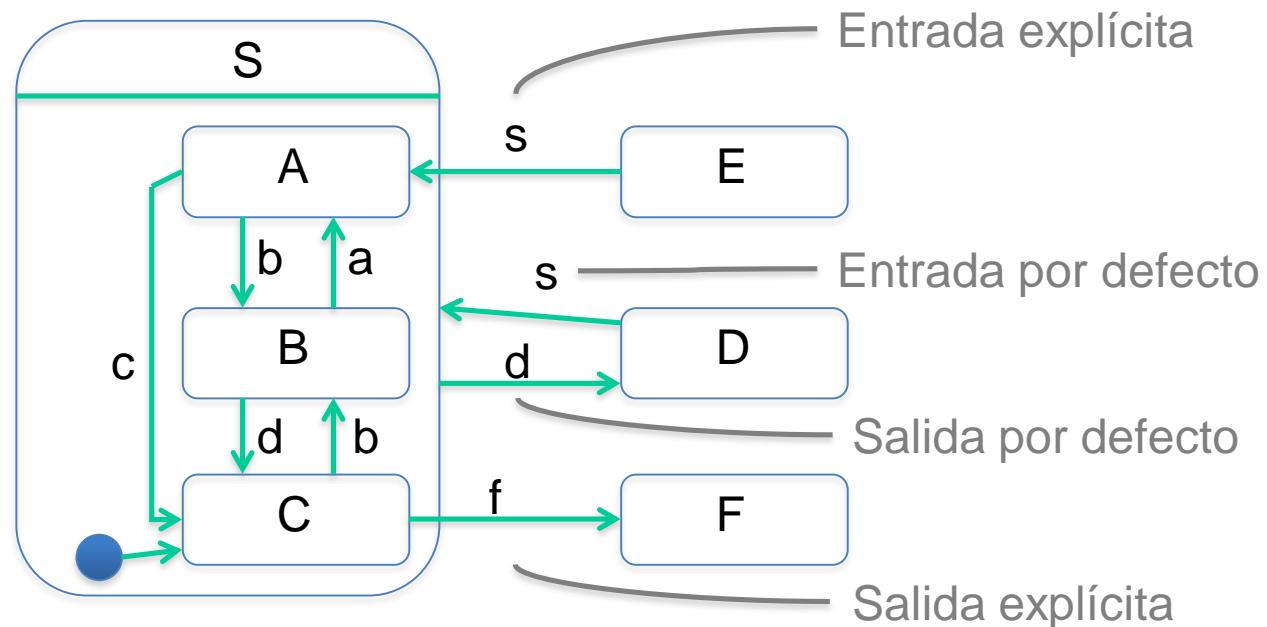


- Ejemplo, redefinición de transición con evento *d*: estando en subestado *B* si se recibe evento *d*, se pasa a subestado *C*. Desde otros subestados, al recibir evento *d* se saldría de superestado *S*.

Estructuración de diagramas ...

□ Generalización ...

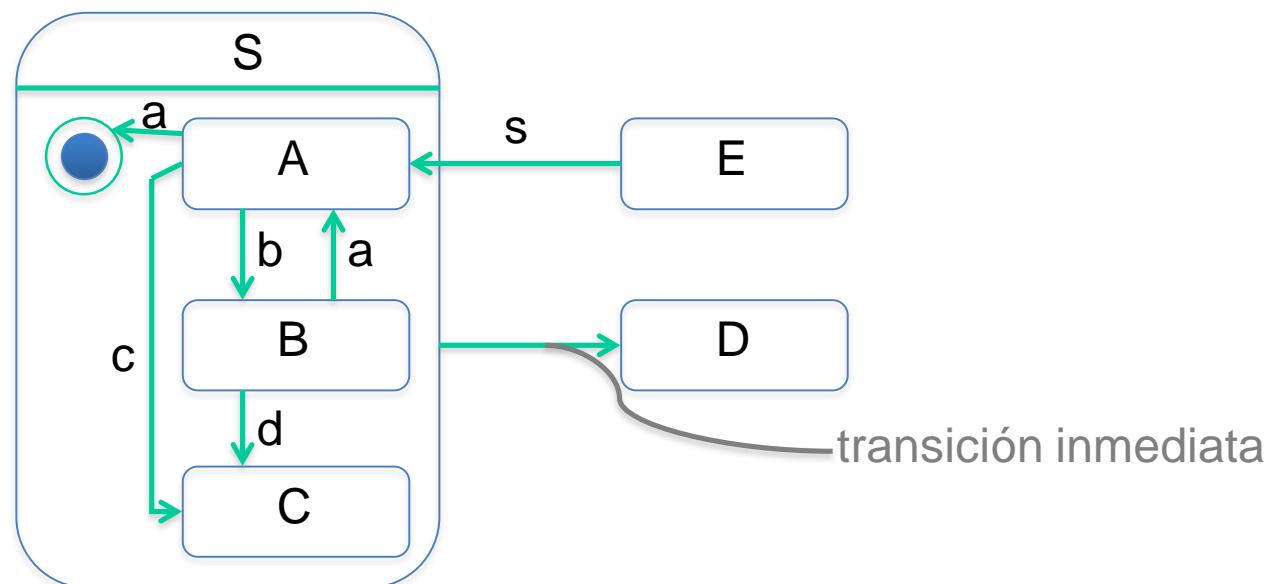
❖ Entrada y salida explícita



Estructuración de diagramas ...

□ Generalización ...

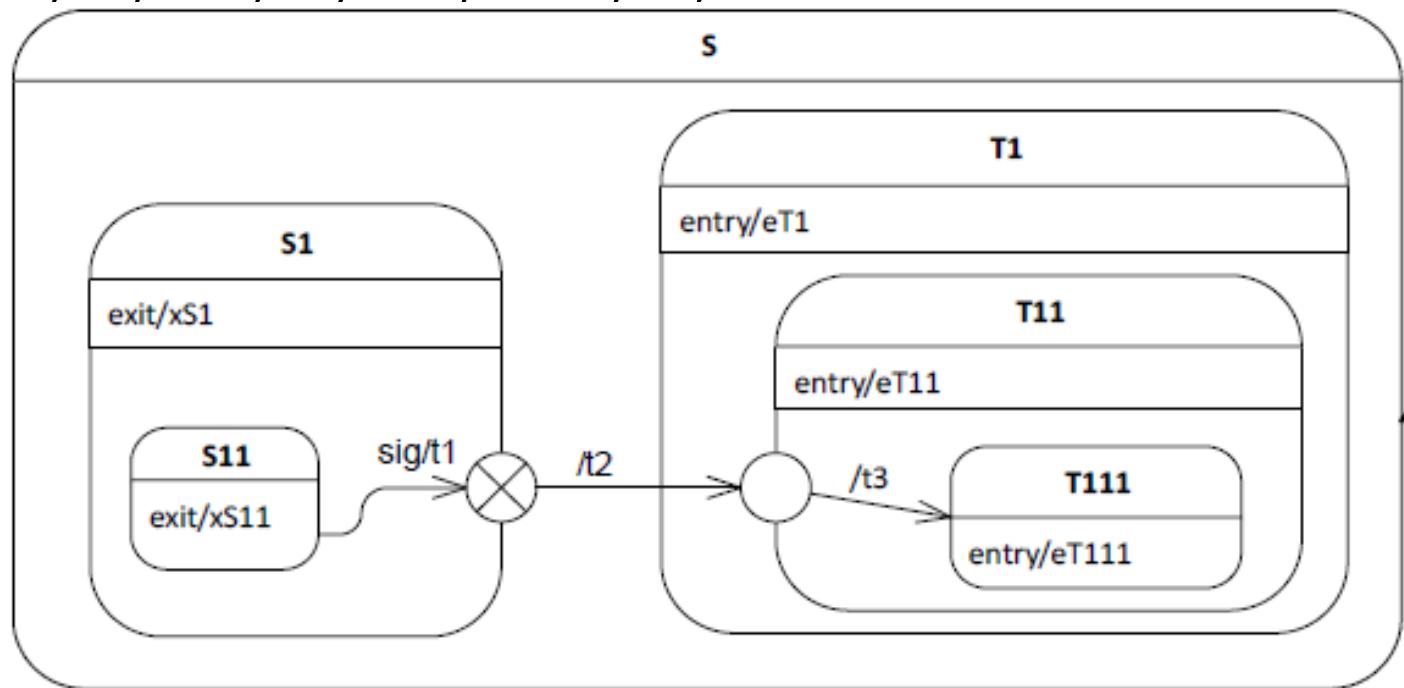
- ❖ Terminación con estado final



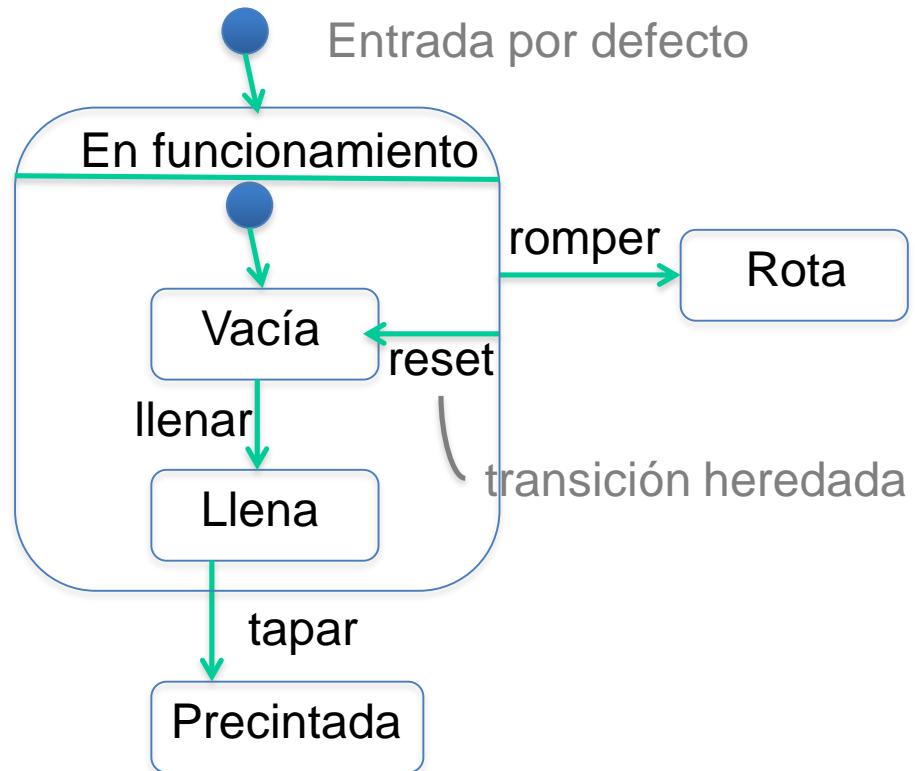
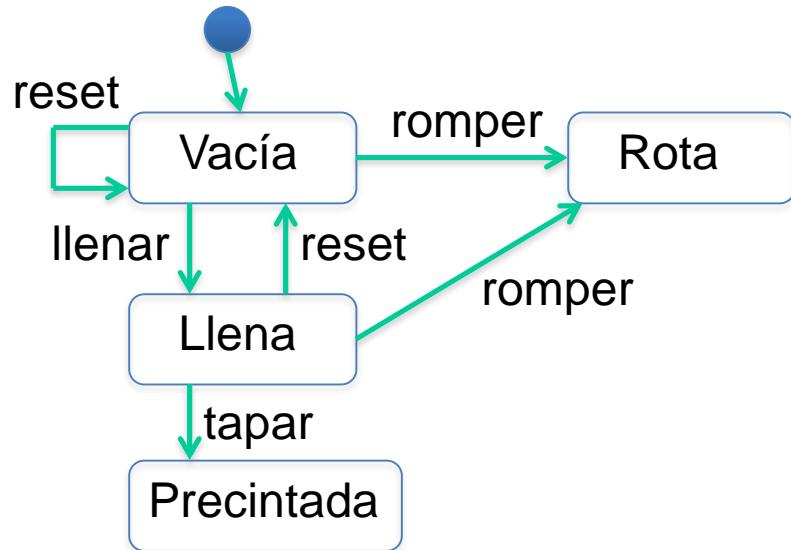
Estructuración de diagramas ...

- Pseudo-estados marcando puntos de entrada y puntos de salida de un estado compuesto
 - ❖ Encapsulan los detalles internos
 - ❖ Secuencia de acciones al dispararse el evento *sig* en el estado S11
 - xS11; t1; xS1; t2; eT1; eT11; t3; eT111

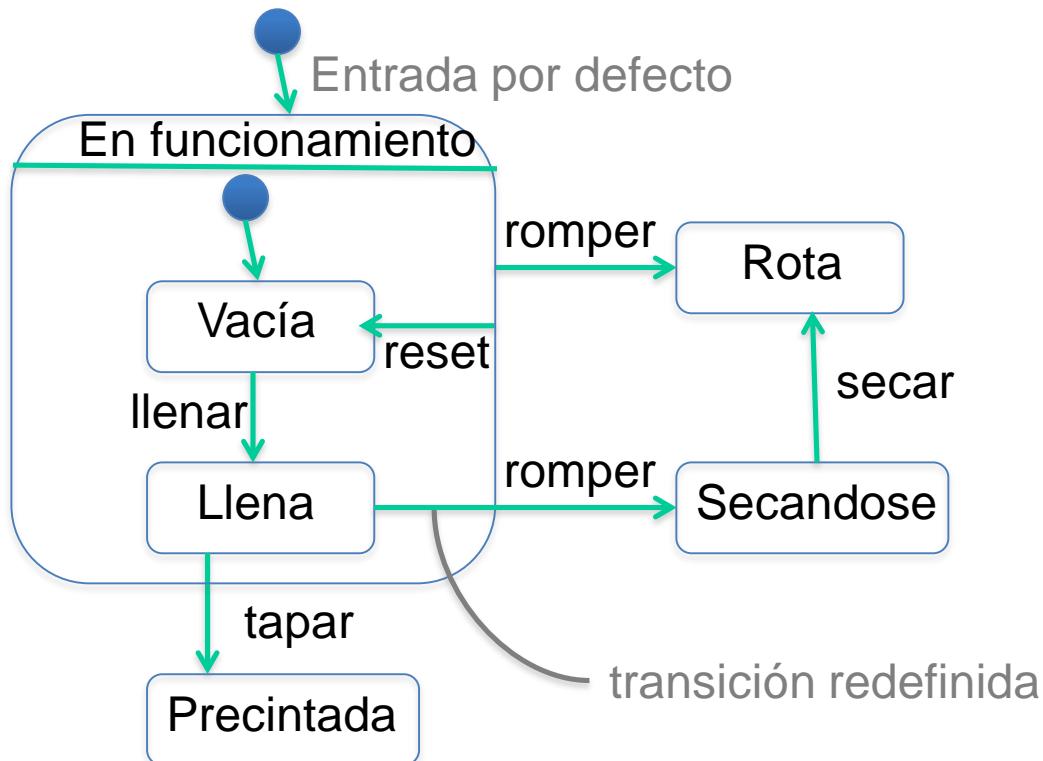
-  Punto de entrada
(entry point)
-  Punto de salida
(exit point)



Ejemplos (estado de la batería de un coche)



Ejemplos (estado de la batería de un coche)



Diagramas de Estados: Concurrencia

□ En el sistema

- ❖ El modelo dinámico describe un conjunto de objetos concurrentes
- ❖ Cada objeto tiene su propio diagrama de estados y por tanto su propio estado
- ❖ Los objetos son concurrentes y cambian de estado independientemente

Concurrencia ...

□ En el sistema ...

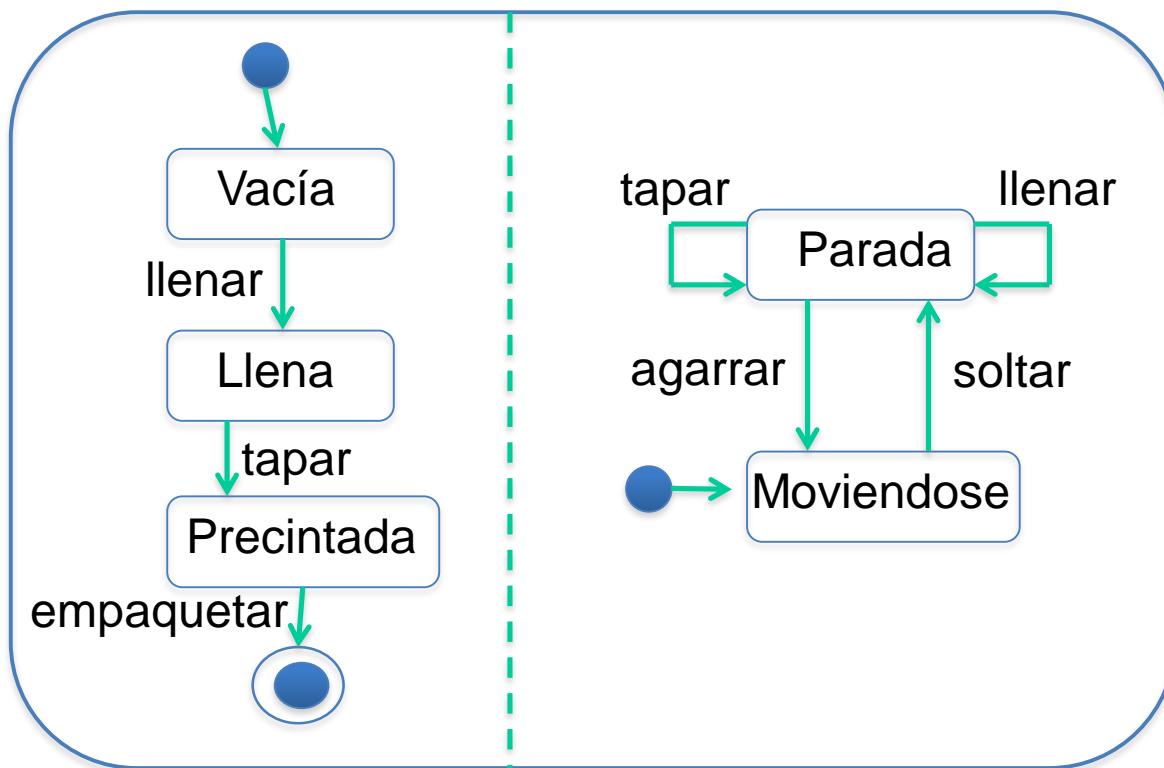
- ❖ El estado del sistema es la *agregación* de los estados de todos los objetos que lo componen
- ❖ Ejemplo: si vemos un coche como un sistema entonces su estado es el estado de la transmisión, el encendido, el acelerador, los frenos, ...

Concurrencia ...

□ Dentro de un objeto

- ❖ La concurrencia dentro del estado de un objeto surge cuando se puede descomponer el objeto en subconjuntos de atributos o de enlaces
- ❖ Cada subconjunto puede definir su propia *región* dentro del diagrama de estados

Ejemplo de diagrama estados con 2 regiones



4. Diagramas de Actividad

- Para sistemas intensivos en software, un diagrama de actividades muestra el flujo de control destacando las actividades que tienen lugar a lo largo del tiempo
- Podemos ver el diagrama de actividades como un diagrama de estados donde no hay eventos y los estados contienen actividades

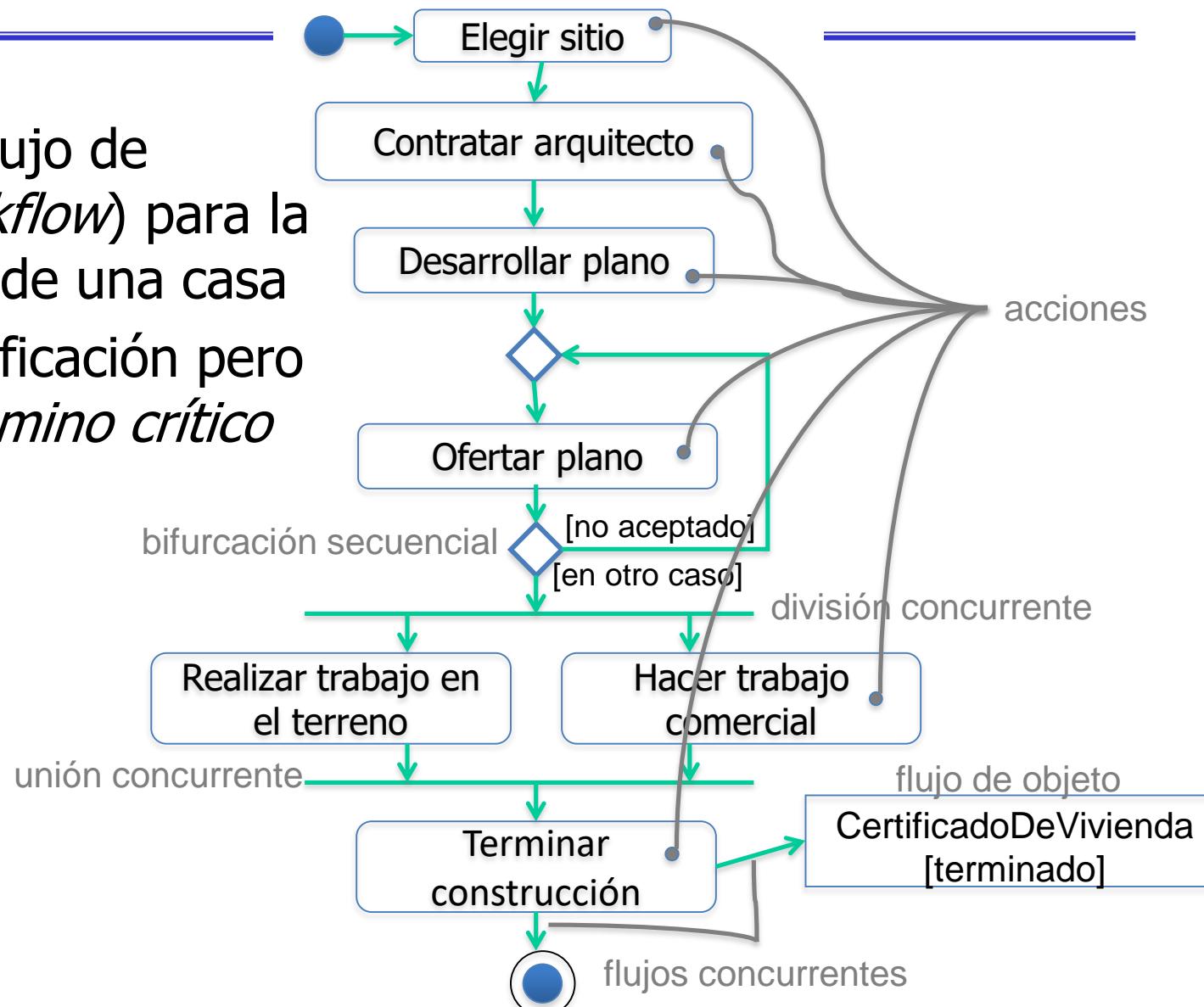
Comparación D. Actividades vs D. Interacción

- ❑ Tanto los diagramas de actividades como los diagramas de interacción permiten modelar un *flujo de trabajo* o el comportamiento dinámico interno de una *operación*
- ❑ Distintas visiones de un mismo comportamiento

D. Interacción	D. Actividades
Los diagramas de interacción se centran en los objetos (recursos)	Los diagramas de actividades se centran en las actividades que tienen lugar entre los objetos
El diagrama de interacción muestra objetos que se pasan mensajes	El diagrama de actividades muestra operaciones que se pasan objetos

Modelado de *workflow*

- ❑ Ejemplo de flujo de trabajo (*workflow*) para la construcción de una casa
- ❑ Es una simplificación pero captura el *camino crítico*



Modelado de operaciones

- ❑ El diagrama de actividades muestra el flujo de *actividades*
- ❑ En el diagrama de estados vimos que una *actividad* era una ejecución no atómica
- ❑ Las actividades ejecutan *acciones* individuales
 - ❖ Producir un cambio de estado en el sistema
 - ❖ Comunicar mensajes
 - ❖ Llamar a otra operación
 - ❖ Cálculos
 - ❖ Crear o destruir objetos, etc.

Diagrama de actividades

- Sintácticamente es un caso especial de Diagrama de Estados donde:
 - ❖ Las transiciones son “disparadas” como consecuencia de la finalización de la acción
 - ❖ Cuando una actividad termina se desencadena el paso a la siguiente actividad
 - ❖ Los estados no poseen transiciones internas ni transiciones desencadenadas por eventos.

Estados de acción

- ❑ Como las acciones del diagrama de estados:
 - ❖ Su ejecución supone un tiempo insignificante en el ámbito del sistema
- ❑ Representan computaciones atómicas:
 - ❖ Crear o borrar un objeto
 - ❖ Invocar una operación sobre un objeto
 - ❖ Evaluar una expresión
 - ❖ Cambiar el valor de un atributo, etc.
- ❑ No se pueden descomponer
- ❑ UML no impone un lenguajes de expresiones

Ofertar plano

indice := buscar(e) + 7



Estados de actividad

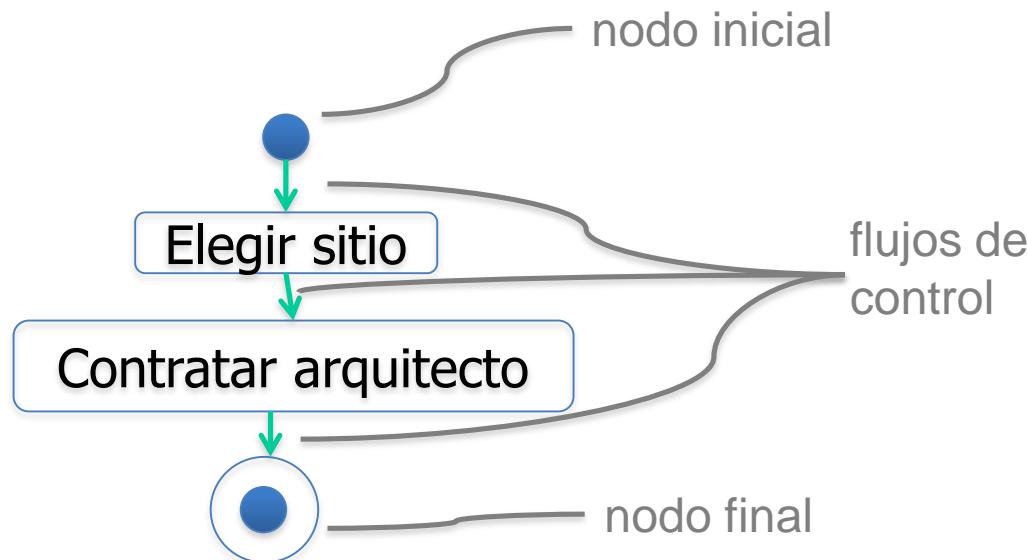
- ❑ Como las actividades del diagrama de estados:
 - ❖ Su ejecución tiene duración
- ❑ Son agrupaciones de acciones o actividades
- ❑ La diferencia con los estados de acción es que se pueden descomponer, es decir, pueden ser especificados por otro D. A.
- ❑ Una herramienta CASE mantendría asociado el estado de actividad y el diagrama que lo especifica

Procesar factura (f)

Construir()

Flujos de control

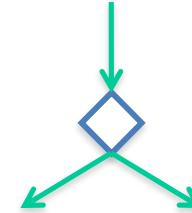
- ☐ Cuando se completa una acción el flujo de control pasa inmediatamente a la siguiente acción
- ☐ El flujo se representa con una flecha sin etiqueta de evento



Nodos de Bifurcación (Decisión) y nodos de Mezcla

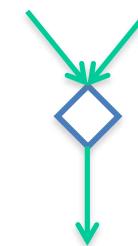
- Permiten modelar iteraciones
- Nodos de bifurcación

- ❖ Representan caminos alternativos
- ❖ Tienen:
 - Una transición de entrada
 - Dos o más de salida (con guardas excluyentes, UML no especifica el lenguaje)
 - Se puede utilizar *else*

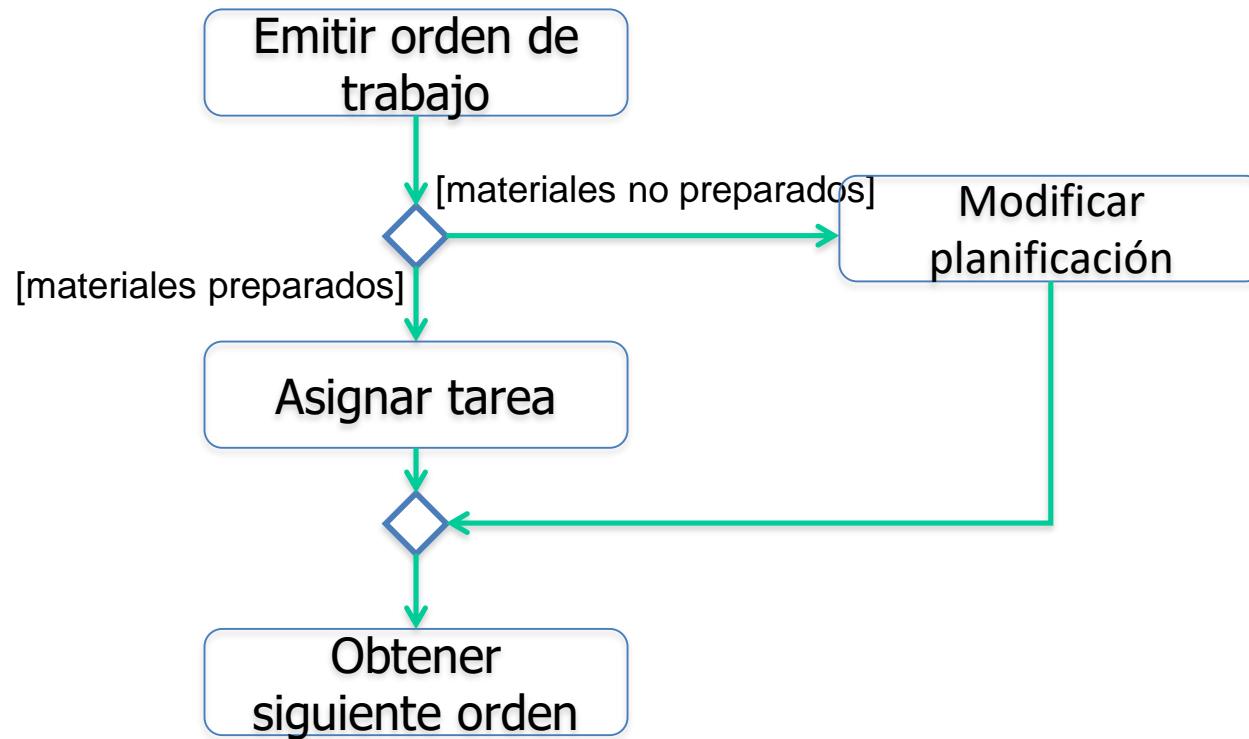


- Nodos de mezcla

- ❖ Permiten mezclar dos o más caminos
- ❖ Tienen:
 - Dos o más transiciones de entrada
 - Y una de salida



Bifurcación ...



Nodos de División y Nodos de Unión

- ❑ Permiten modelar flujos concurrentes

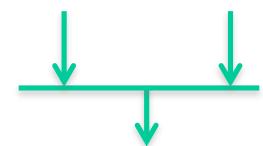
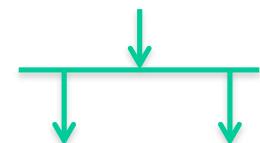
- ❑ Nodos de división:

- ❖ Una transición de entrada y dos o más de salida
 - ❖ Los caminos continúan en paralelo

- ❑ Nodos de unión:

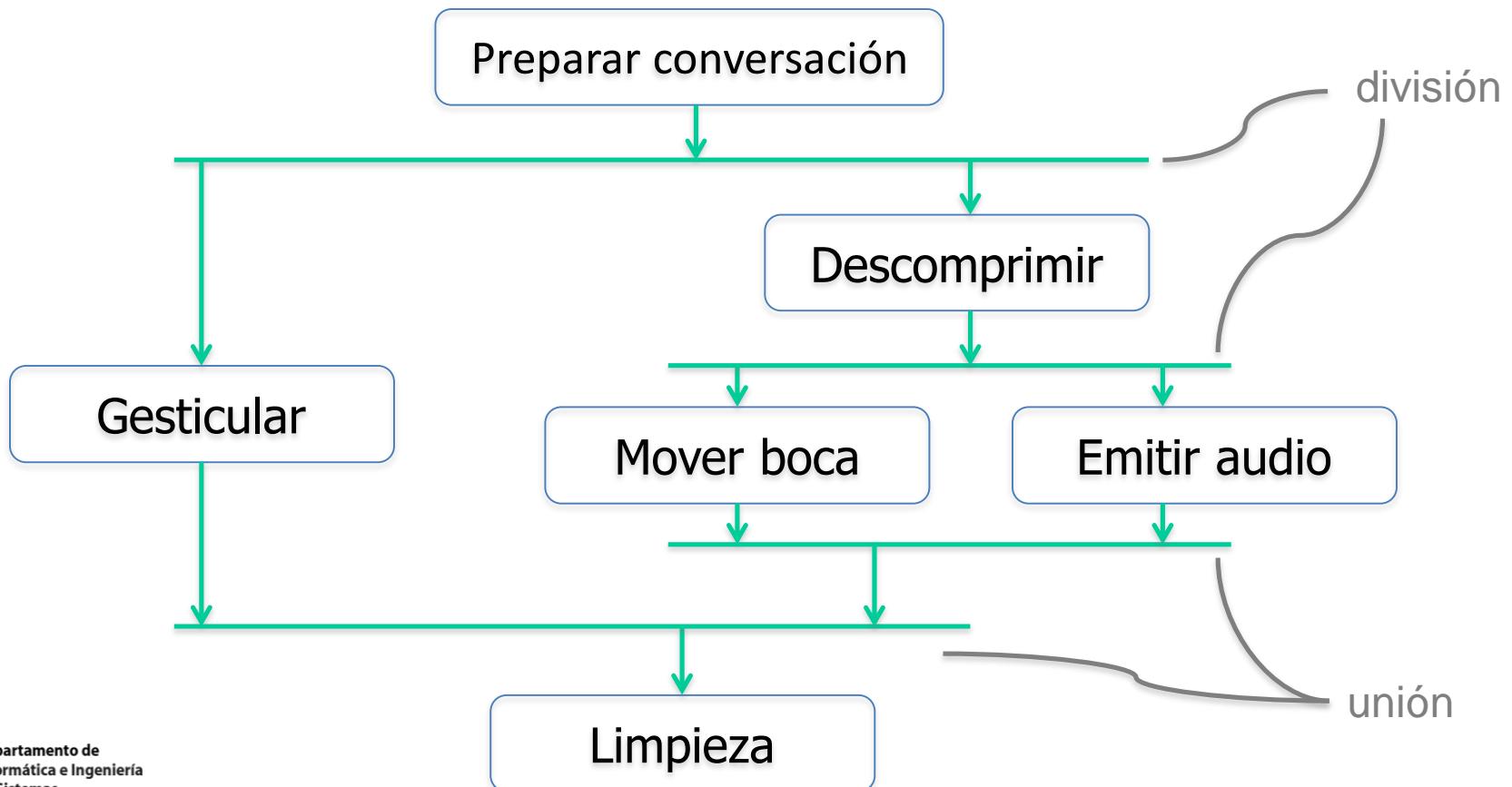
- ❖ Dos o más transiciones de entrada.
 - ❖ Una transición de salida.

- ❑ La unión sincroniza. Cada flujo espera hasta que todos los flujos de entrada alcanzan la unión



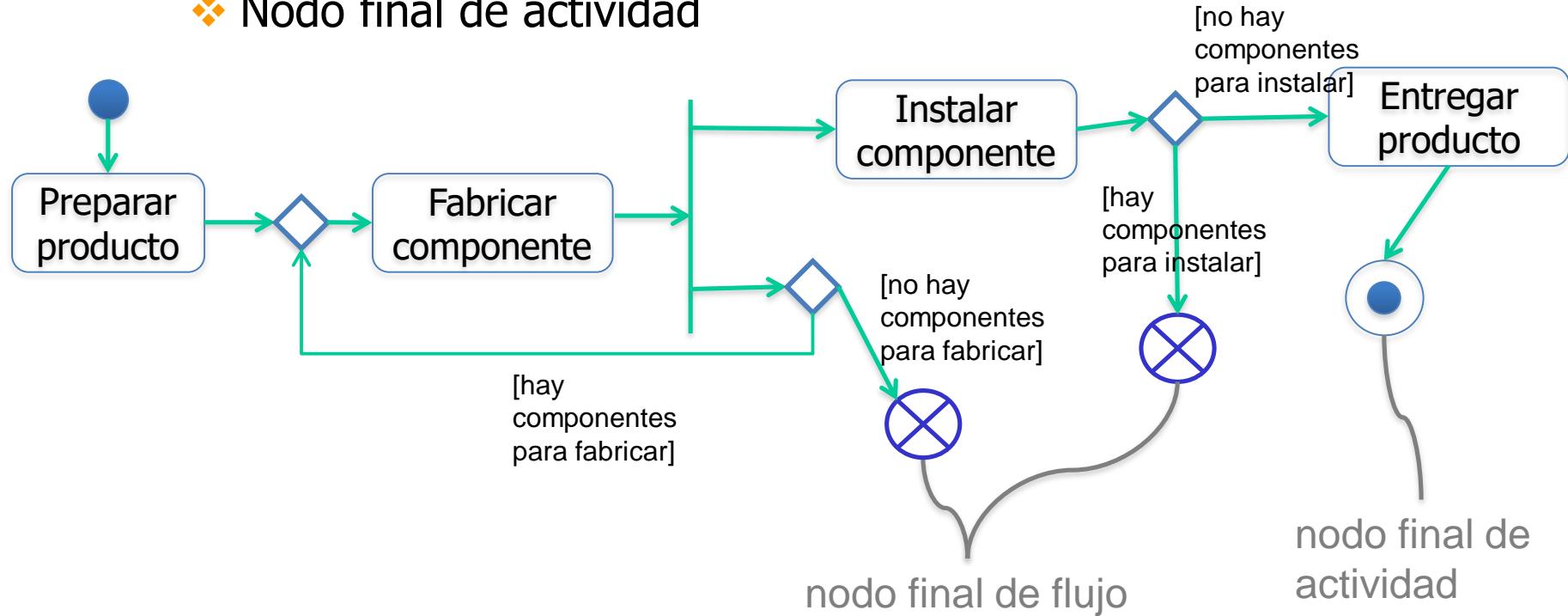
División y unión ...

Ejemplo de flujo de trabajo para reproducir una conversación en un dispositivo que imita la voz y los gestos humanos:



Dos tipos de nodos finales

- En un flujo de control puede haber dos tipos de nodos finales
 - ❖ Nodo final de flujo
 - ❖ Nodo final de actividad



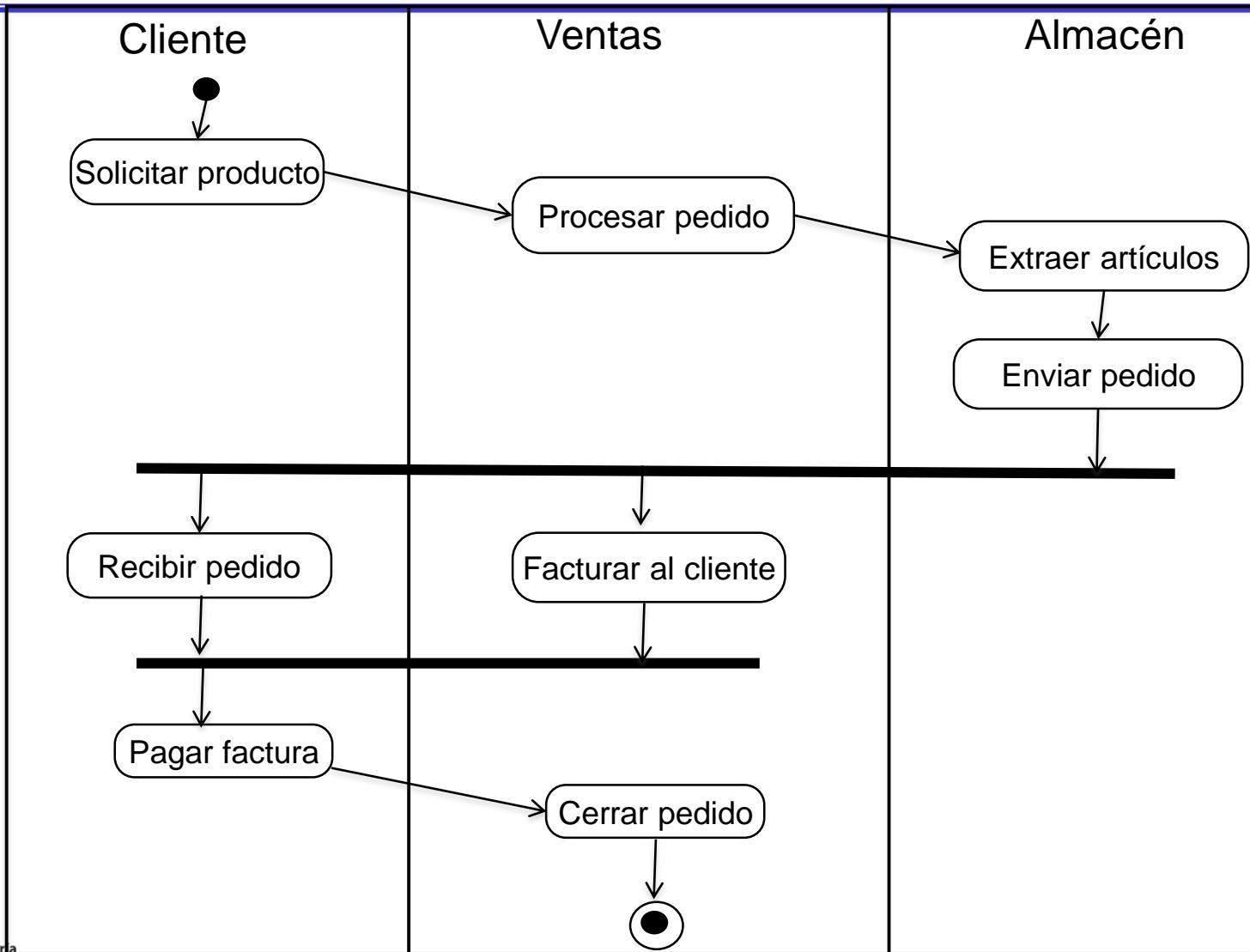
nodo final de flujo

nodo final de actividad

Calles

- ❑ Particionan los nodos en grupos
- ❑ Cada grupo o calle representa la parte de la organización responsable de llevar a cabo las actividades
- ❑ Cada calle tiene un nombre único
- ❑ Una calle podría ser implementada por varias clases
- ❑ Los flujos pueden cruzar las calles

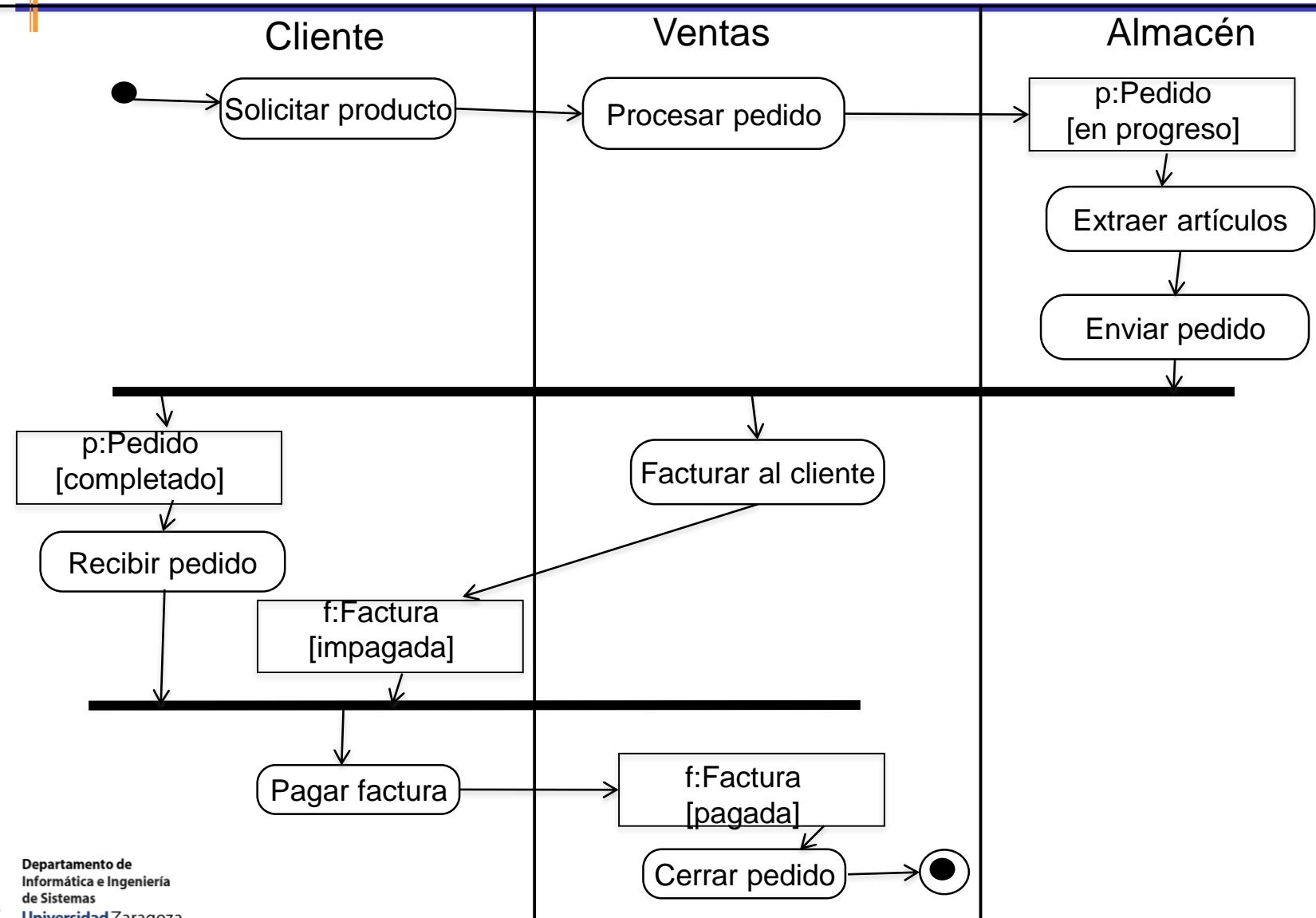
Calles ...



Flujo de objetos

- En el flujo de control pueden verse involucrados objetos. En el ejemplo anterior la *factura* y el *pedido*
- Algunas actividades crean esos objetos, otras los usan o modifican
- También se representa el estado del objeto

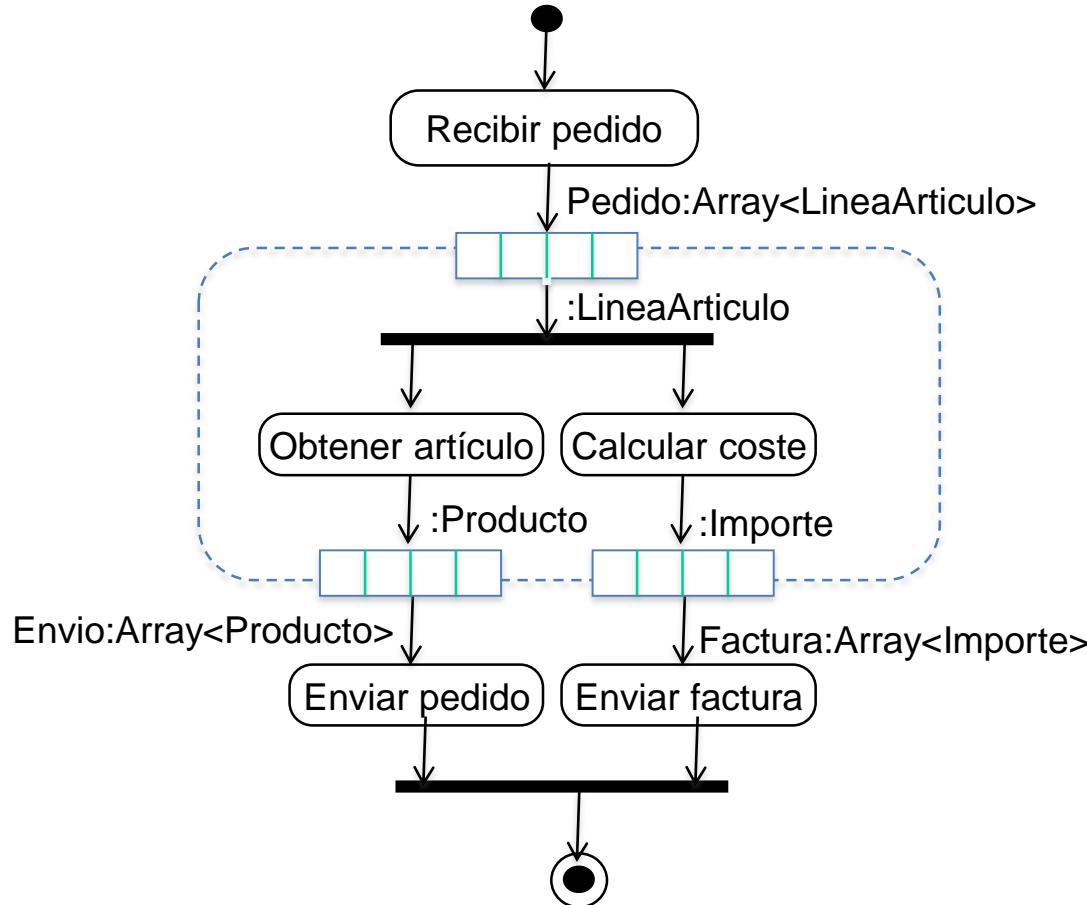
Flujo de objetos ...



Regiones de expansión (I)

- Se usan cuando una misma operación se ejecuta sobre un conjunto de elementos → para evitar modelar con iteraciones (oscurecen el significado de la operación)
- Formato:
 - ❖ Las entradas y las salidas son colecciones de valores → se representan con un *array*
 - ❖ Los elementos individuales del *array* se ejecutan en la región (si es posible concurrentemente)

Regiones de expansión (II)



Usos del diagrama

- En general para modelar la dinámica de cualquier elemento: sistema, subsistema, clase, caso de uso, etc.
- Los usos más habituales:
 - ❖ Modelar flujos de trabajo
 - A menudo, el sistema que vamos a desarrollar se integra en un entorno con gran cantidad de software (sistemas intensivos en software) donde existen flujos de trabajo
 - Los D.A. hacen hincapié en las actividades tal y como son vistas por los actores del sistema
 - modelan los procesos de negocio donde colaboran humanos y sistemas automáticos
 - ❖ Modelar operaciones
 - Los D.A. se utilizan como diagramas de flujo para modelar los detalles de una computación

Modelado de flujos de trabajo ...

- Es imposible mostrar en un solo diagrama todos los flujos de trabajo → centrarse en un centro de interés
- Establecer los límites del flujo → precondiciones y poscondiciones
- Desde el estado inicial especificar las acciones que tienen lugar a lo largo del tiempo
- Estructurar las actividades
- Considerar bifurcaciones, concurrencia
- Especificar el flujo de objetos