INGENIERÍA DEL SOFTWARE I

Tema 10

Comportamiento del Sistema

(en desarrollo OO)

Univ. Cantabria – Fac. de Ciencias

Francisco Ruiz, Juan Hernández

Objetivos del Tema y Bibliografía

- Conocer los conceptos básicos (eventos, máquinas de estados y actividades) para el modelado avanzado del comportamiento de los objetos y el sistema.
- Conocer las características de los diagramas de estados y de actividades.
- Aprender a utilizar dichos diagramas para modelado del comportamiento.
- Bibliografía Básica
  - Booch, Rumbaugh y Jacobson (2006): El Lenguaje Unificado de Modelado. 2ª edición. Caps. 20-22 y 25.
- Bibliografía Complementaria
  - Rumbaugh, Jacobson y Booch (2007): El Lenguaje Unificado de Modelado. Manual de Referencia. 2ª edición. Caps. 7 y 8.
  - Miles y Hamilton (2006): Learning UML 2.0. Caps. 3 y 14.

Contenido

- Introducción
- Eventos
  - Tipos
  - Envío y Recepción
- Diagramas de Estados
  - Máquinas de Estados
  - Estados
  - Transiciones
  - Tipos de Estados
  - Estados Compuestos
  - Representación de Diagramas de Estados
  - Consejos
- Diagramas de Actividades
  - Tipos de Acciones
  - Contenido
  - Flujo de Control
  - Flujo de Objetos
  - Expansiones
  - Consejos
- Apéndice A: Modelado
  - Flujo de Trabajo
  - Operación
  - Familia de Señales
  - Excepciones
  - Objeto Reactivo

Introducción

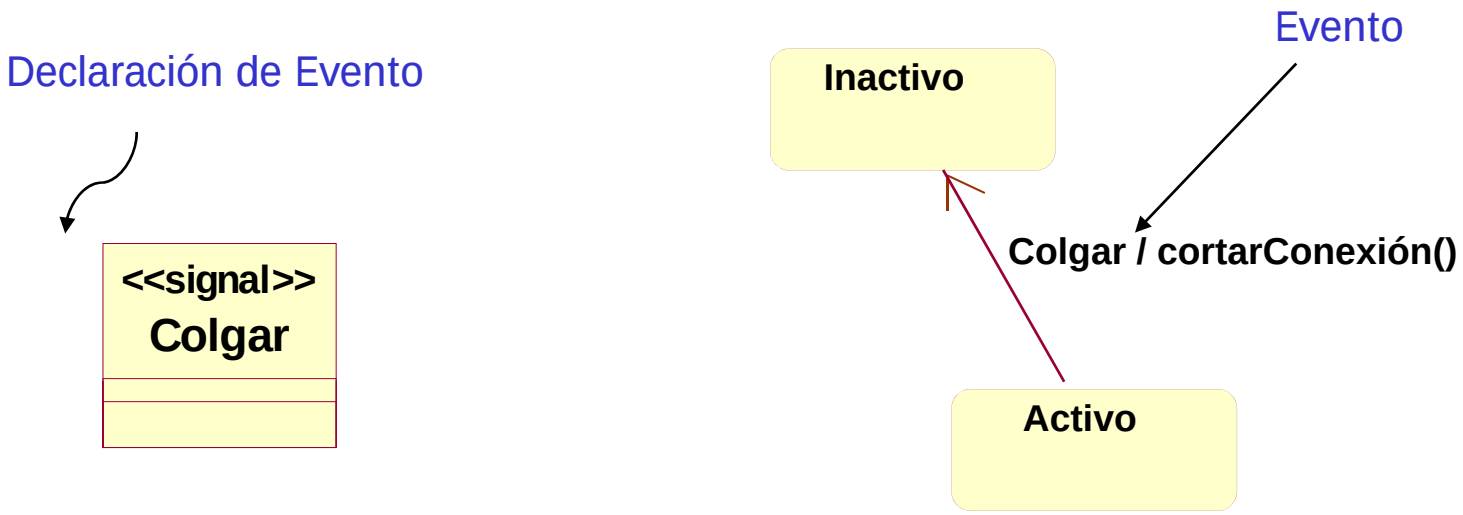
- En el tema 7 se estudiaron las interacciones, como mecanismos de modelado de aspectos dinámicos de un sistema.
- A continuación se incluyen el resto de conceptos y diagramas empleados para modelar el **comportamiento dinámico de un sistema**, determinado por cosas que ocurren externa o internamente (eventos).
- El comportamiento de un objeto individual se modela mediante una **máquina de estados**.
  - El comportamiento de una sociedad de objetos ya se estudió (colaboraciones).
- Además de los diagramas de casos de uso, y de interacción (Secuencia y Comunicación), para modelar aspectos dinámicos de un sistema también se usan:
  - **Diagramas de estados** (orientados a eventos), y
  - **Diagramas de actividades** (orientados a actividades).

Eventos

- Un **evento** es la especificación de un acontecimiento significativo, ubicado en el tiempo y en el espacio.
- Se utilizan en máquinas de estado para modelar la aparición de un estímulo que puede disparar la transición de un estado a otro.
- Pueden ser:
  - **Síncronos**
    - Llamadas (invocación de operaciones, ya vistos)
  - **Asíncronos**
    - Señales (excepciones), Paso de tiempo, Cambio de estado.

Eventos

- Notación en UML 2



- *Declaración de un evento Colgar de tipo señal.*
- *El evento Colgar produce un cambio desde el estado Activo al estado Inactivo.*
- *Además, se lleva a cabo la acción cortarConexión.*

Eventos – Tipos

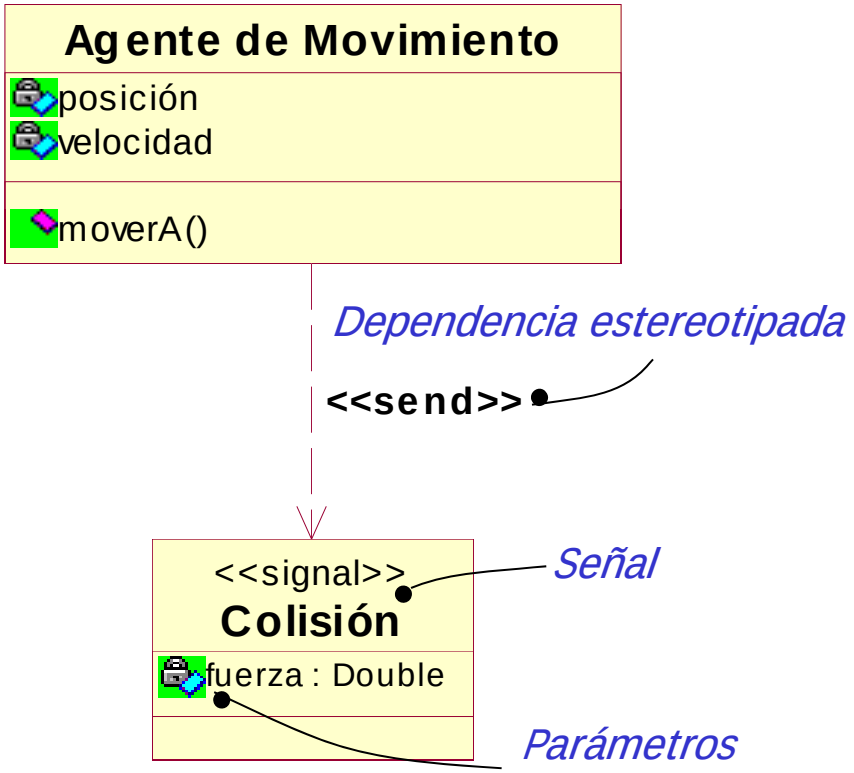
- En cuanto a dónde acontecen, pueden ser:
  - **Externos**, si fluyen entre el sistema y sus actores (pulsación del ratón).
  - **Internos**, si fluyen entre objetos del sistema (una excepción).
- En UML 2 se pueden modelar cuatro **clases de eventos**:
  - De **Señal**: Recepción de una comunicación asíncrona, explícita y con nombre, entre objetos.
  - De **Llamada**: Recepción, por un objeto, de una petición explícita síncrona.
  - De **Tiempo**: Llegada de un tiempo absoluto o transcurso de una cantidad relativa de tiempo.
  - De **Cambio**: Un cambio en el valor de una expresión booleana.

Eventos – Tipos

- **Señales**.
  - Representa la acción de enviar (lanzar) de manera asíncrona un objeto para que otro lo reciba
    - Ejemplo: Excepciones (tipo más común de señal interna).
  - Tienen mucho en común con las clases:
    - Pueden tener instancias, atributos (parámetros) y operaciones, aunque no se suelen modelar explícitamente.
    - También pueden existir relaciones de generalización entre señales (jerarquías de señales).
  - Origen de una señal:
    - Acción de una transición de un estado a otro.
    - Envío de un mensaje entre dos roles en una interacción.
    - Ejecución de una operación.

Eventos – Tipos

- **Señales**.
  - En UML 2 las señales se modelan como clases estereotipadas con «signal».
  - Para indicar que una operación envía una señal se puede utilizar una dependencia estereotipada como «send».

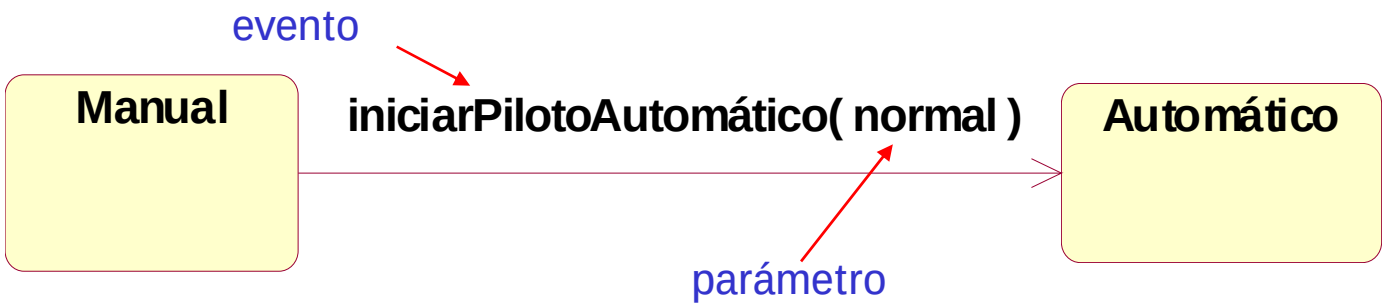


Eventos – Tipos

- Un evento de **Llamada**.
  - Representa la invocación de una operación.
  - Suele ser síncrono.
    - Cuando se invoca la operación se queda a la espera de recibir retorno.
  - Dos casos especiales son el evento de creación de un objeto (*«create»*) y el de su destrucción (*«destroy»*).
  - En una máquina de estados implica los siguientes pasos:
    1. El control pasa del emisor al receptor.
    2. El evento dispara una transición.
    3. La operación acaba.
    4. El receptor pasa a un nuevo estado.
    5. El control regresa al emisor.

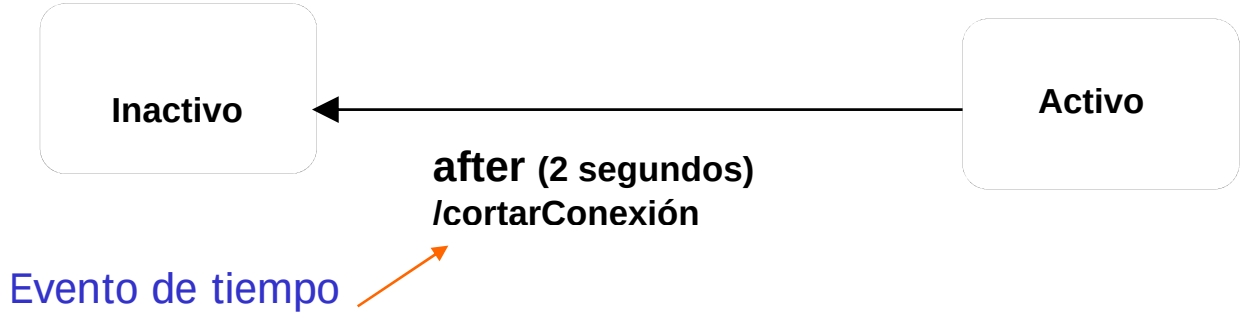
Eventos – Tipos

- Un evento de **Llamada**.
  - El modelado en UML 2 de un evento de llamada es similar al de una señal.
    - Se muestra el evento con sus parámetros, como el disparador de una transición de estado.



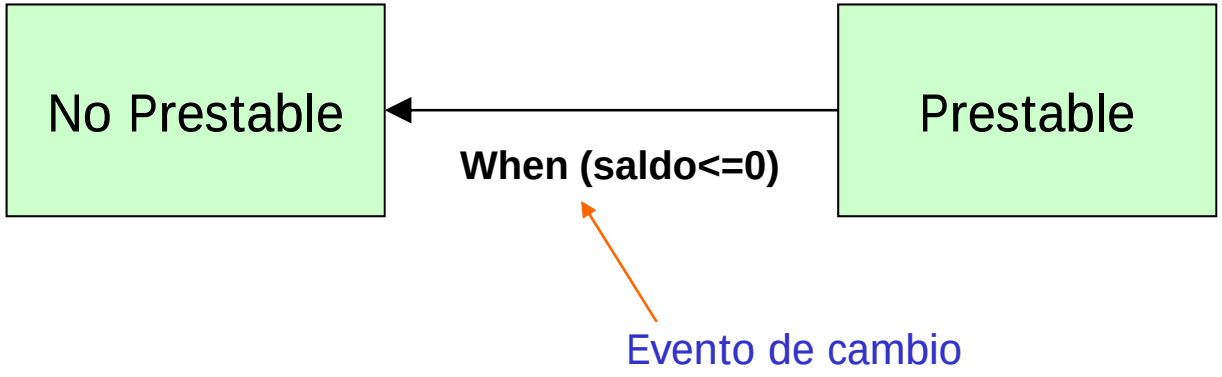
Eventos – Tipos

- Un evento **de Tiempo** representa un instante en el tiempo mediante una expresión.
- La expresión puede ser
  - **Absoluta**: at (<valor temporal>)
    - Ejemplo: at (21:00h)
  - **Relativa**: after (<valor temporal>)
    - Debe usarse en el contexto de un disparador (trigger) de forma que el tiempo 0 es el de inicio de actividad del disparador.
    - Por defecto el disparador es la entrada en el estado actual.



Eventos – Tipos

- Un evento **de Cambio** representa un cambio de estado o el cumplimiento de alguna condición.
  - Notación: When (<expresión booleana>)
  - Ocurre una vez cuando el valor de la expresión cambia de falso a verdadero.
    - No cuando pasa de cierta a falsa.
    - No se repite mientras la expresión sigue siendo cierta.

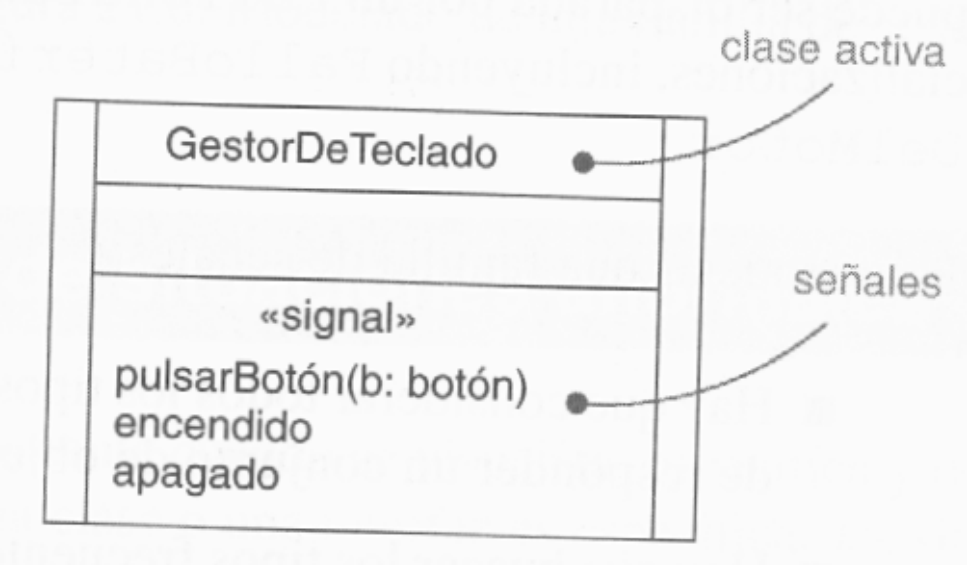


Eventos – Envío y Recepción

- Los eventos de señal o llamada implican al menos dos objetos:
  - **Emisor** (el que envía la señal o invoca la operación), y
  - **Receptor**.
- Cualquier instancia de cualquier clase puede:
  - Enviar una señal a un objeto receptor.
  - Invocar (llamar) una operación de un objeto receptor.
  - Recibir un evento de llamada o una señal desde un objeto emisor.
- Envíos múltiples:
  - **Multicasting**: un objeto envía una señal a una colección de objetos.
  - **Broadcasting**: Un objeto envía una señal a cualquier objeto del sistema que esté “escuchando” (el sistema se modela como otro objeto).

Eventos – Envío y Recepción

- En UML 2
  - Los eventos de llamada que puede recibir un objeto se modelan como operaciones sobre la clase del objeto.
  - Las señales con nombre que puede recibir un objeto se modelan designándolas en un compartimento extra de la clase.



Máquina de Estados

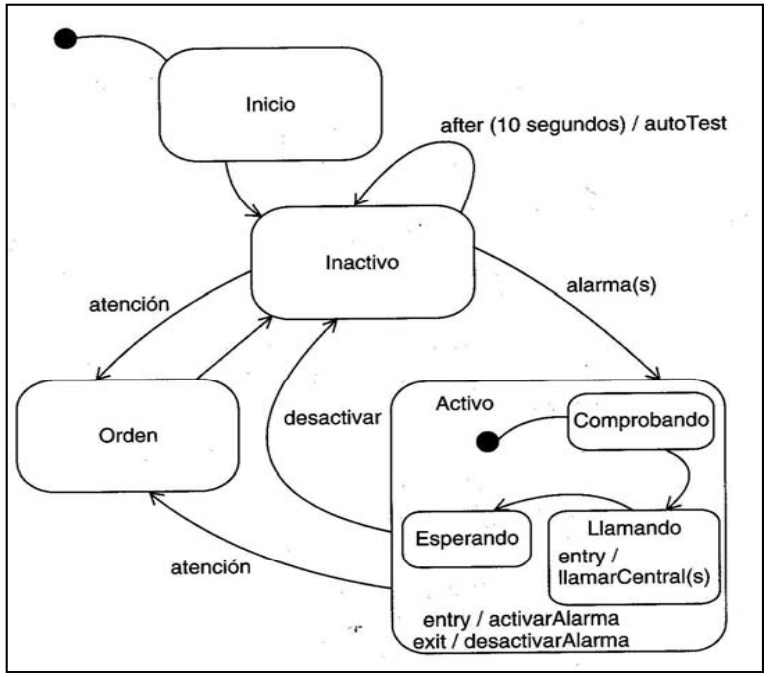
- Una **Máquina de Estados** es un comportamiento que especifica las secuencias de estados por las que pasa **un objeto** a lo largo de su vida en respuesta a eventos, junto con sus respuestas a dichos eventos.
  - La mayoría de las veces esto implica modelar la vida de las instancias de una clase, un caso de uso o un sistema completo.
- Sirven para modelar aspectos dinámicos de un sistema.
  - Comportamiento de UN OBJETO. El de una sociedad de objetos lo modelan las interacciones.
  - Comportamiento de sistemas completos, especialmente reactivos, que deben responder a señales de actores externos al sistema.
  - Comportamiento de un caso de uso (aunque la mayoría de las veces se hace con interacciones).

Máquina de Estados

- Son la mejor forma de especificar el comportamiento de los **objetos reactivos**, es decir, instancias que
  - deben responder a eventos externos o internos,
  - tienen un comportamiento que depende de su historia, y
  - tienen un ciclo de vida modelado como una progresión de estados, transiciones y eventos.
- Una máquina de estados puede visualizarse de dos formas:
  - Destacando los estados de los objetos y sus transiciones (**Diagramas de Estados**).
  - Destacando el flujo de control entre actividades (**Diagramas de Actividades**).

Máquina de Estados

- Una máquina de estados tiene forma de grafo dirigido con diferentes tipos de arcos y de nodos.
  - Los **nodos** representan los distintos **estados** por los que puede pasar un objeto.
  - Los **arcos** se corresponden con las **transiciones** entre dichos estados. Normalmente, estas transiciones serán disparadas por **eventos**.

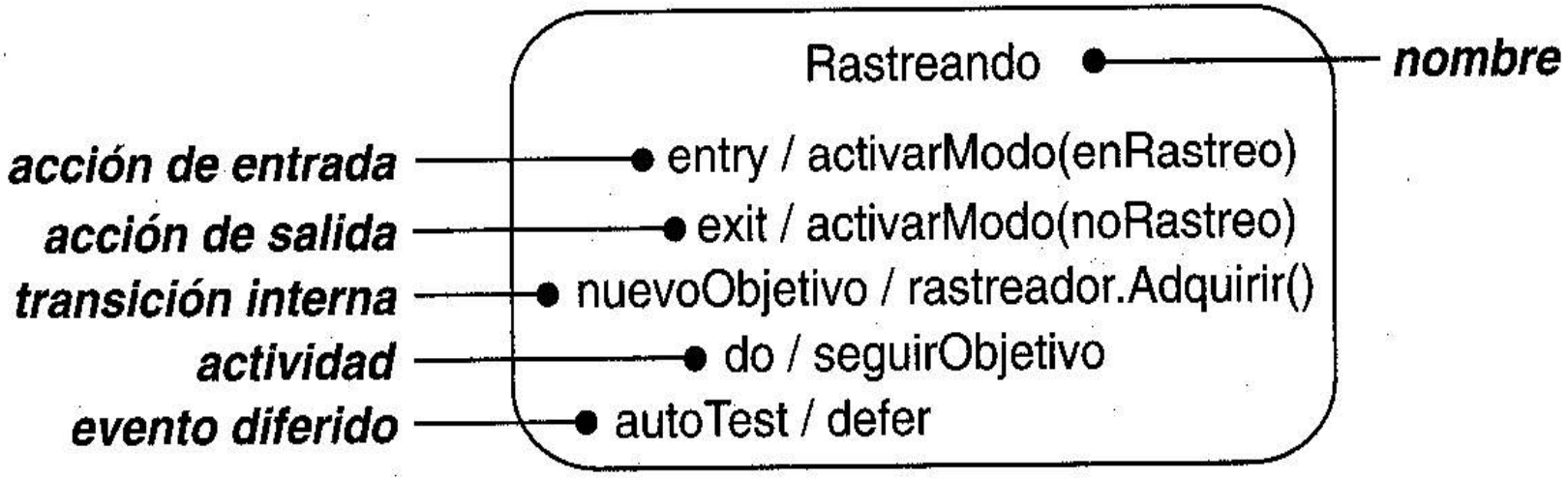


Máquina de Estados - Estados

- Un **Estado** es una condición o situación en la vida de un objeto durante la cual el objeto satisface alguna condición, realiza alguna actividad o espera algún evento.
  - Un objeto permanece en un estado durante una cantidad finita de tiempo.
- **Partes** de un estado:
  - **Nombre**. Identificador (puede haber estados anónimos).
  - **Acciones de E/S**. Se ejecutan al entrar o salir del estado.
  - **Transiciones internas** (autotransiciones). Se manejan sin causar un cambio de estado.
  - **Actividades (Do)**. Actividades que se realizan mientras el estado está activo.
  - **Eventos diferidos**. Lista de eventos que se posponen para ser manejados en otro estado.

Máquina de Estados - Estados

- Con las acciones de entrada/salida, las transiciones internas y los eventos diferidos es posible modelar comportamientos complejos.
  - Estas características se representan como cadenas de texto dentro de un compartimento en el icono del estado.

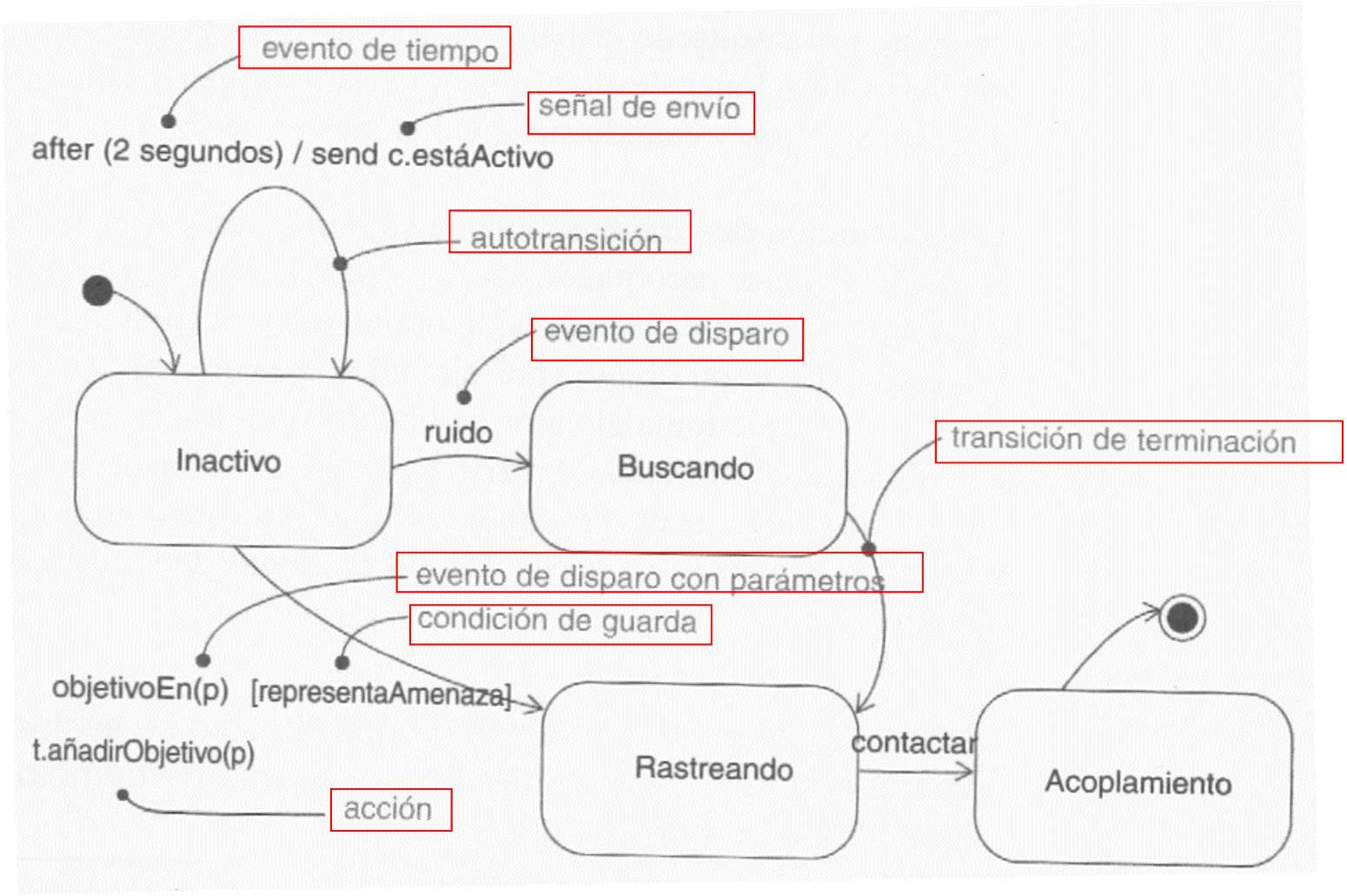


Máquina de Estados - Transiciones

- Una **Transición** es una relación entre dos estados que indica que un objeto, que esté en el primer estado, realizará ciertas acciones y entrará en el segundo estado cuando ocurra un evento especificado y se satisfagan unas condiciones determinadas.
- Una transición tiene dos **partes** obligatorias:
  - **Estado origen**. Estado activo antes de iniciar la transición.
  - **Estado destino**. El estado activo cuando finaliza la transición.
- Y tres opcionales:
  - **Evento de disparo**. Provoca el disparo de la transición.
  - **Condición de guarda**. Expresión booleana. Sólo si es verdadera se puede disparar la transición.
  - **Acción**. Comportamiento ejecutable (acción atómica) que puede actuar directamente sobre el objeto asociado a la máquina de estados, e indirectamente sobre otros.

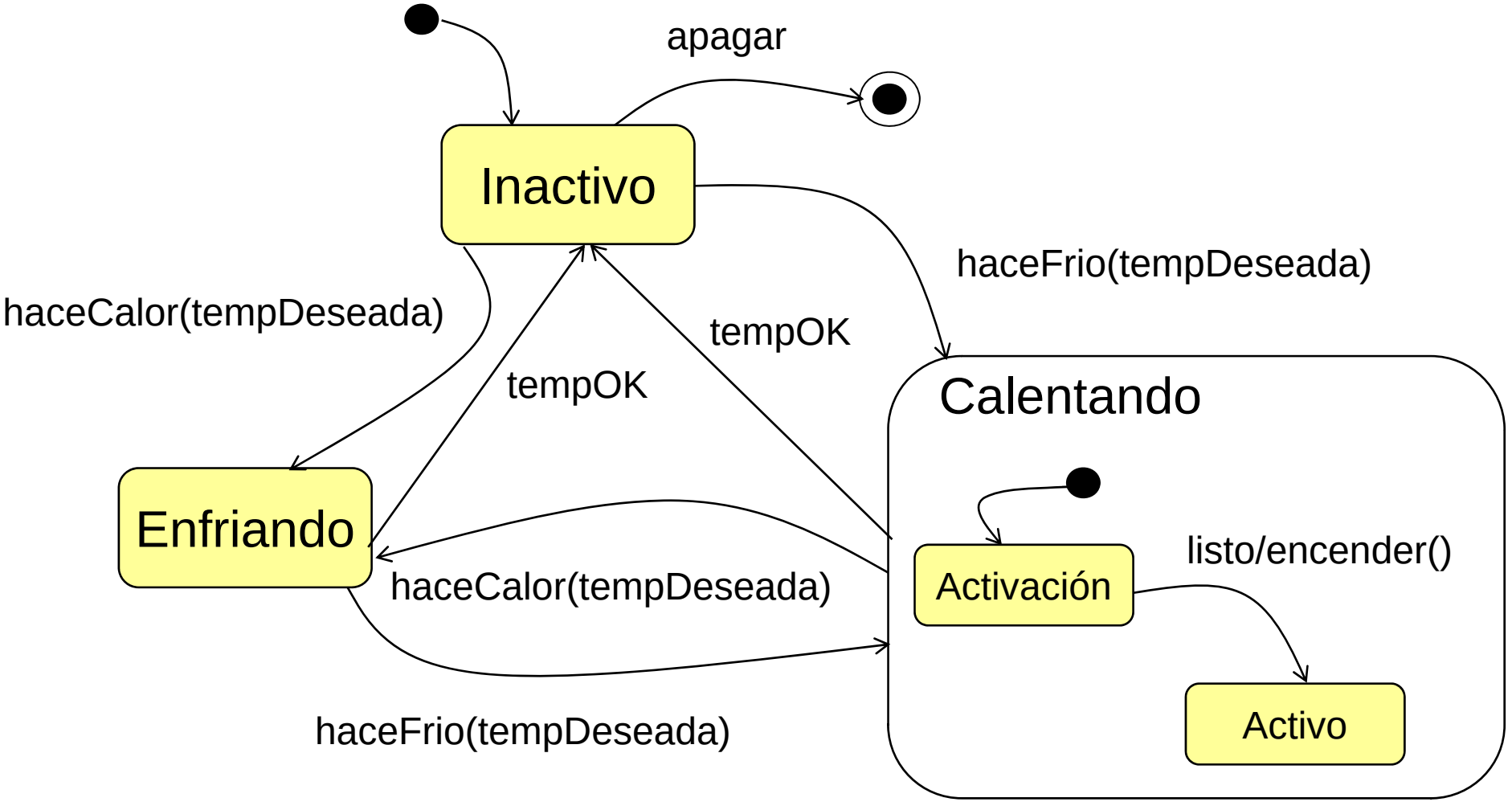
Máquina de Estados - Transiciones

- **Transiciones**.

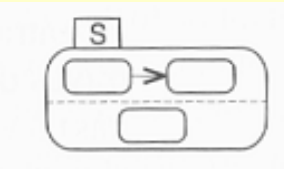
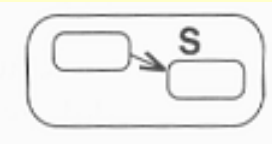


Máquina de Estados

- **Ejemplo** de un Termostato.

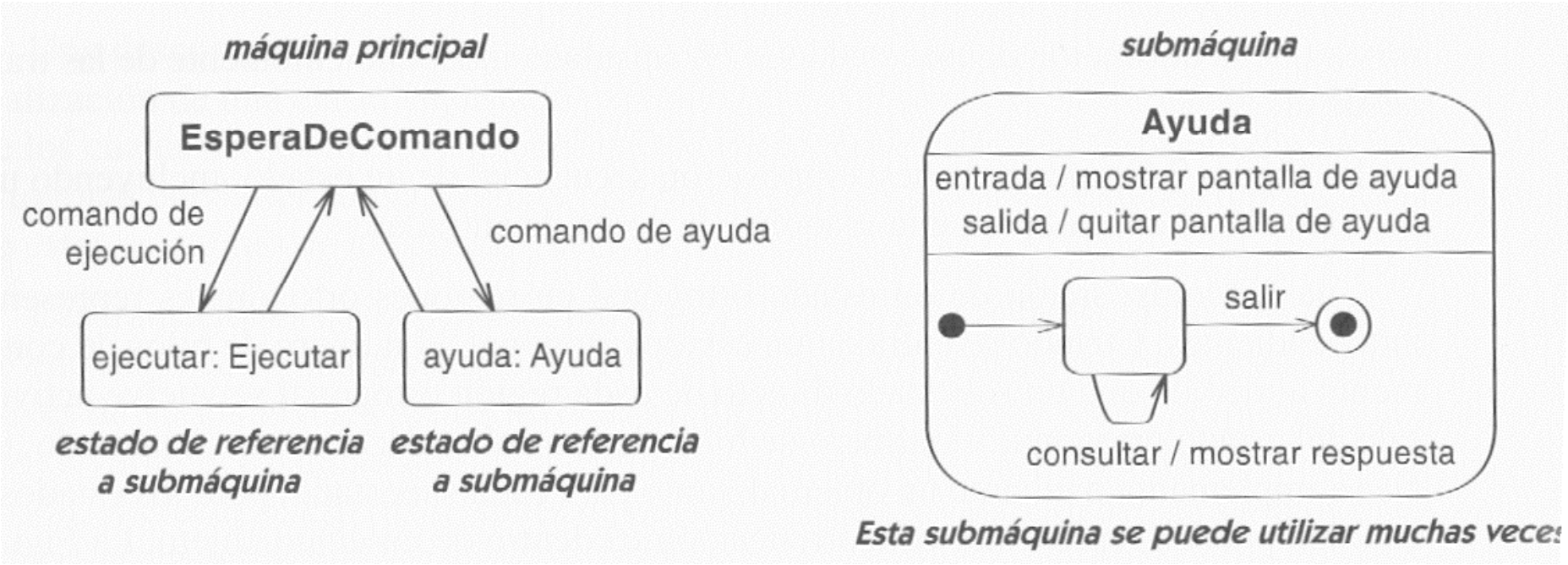


Máquina de Estados - Tipos

Tipo de Estado	Descripción	Notación
Simple	Sin estructura interna.	
Compuesto El estado tiene estructura interna con varios estados interiores (subestados).	**Estado Ortogonal** (subestados concurrentes): Se divide en dos o más regiones. Cuando el estado está activo significa que lo está uno de los subestados de cada región.	
	Estado No Ortogonal (subestados secuenciales): Contiene uno o más subestados directos. Cuando el estado está activo significa que lo está uno y solo uno de los subestados.	
Submáquina	Semánticamente, un estado submáquina es equivalente a un estado compuesto. Se utiliza para factorizar comportamiento (cuando el mismo comportamiento debe aparecer en varios lugares). Una submáquina puede ser referenciada desde dentro de otras máquinas de estado.	

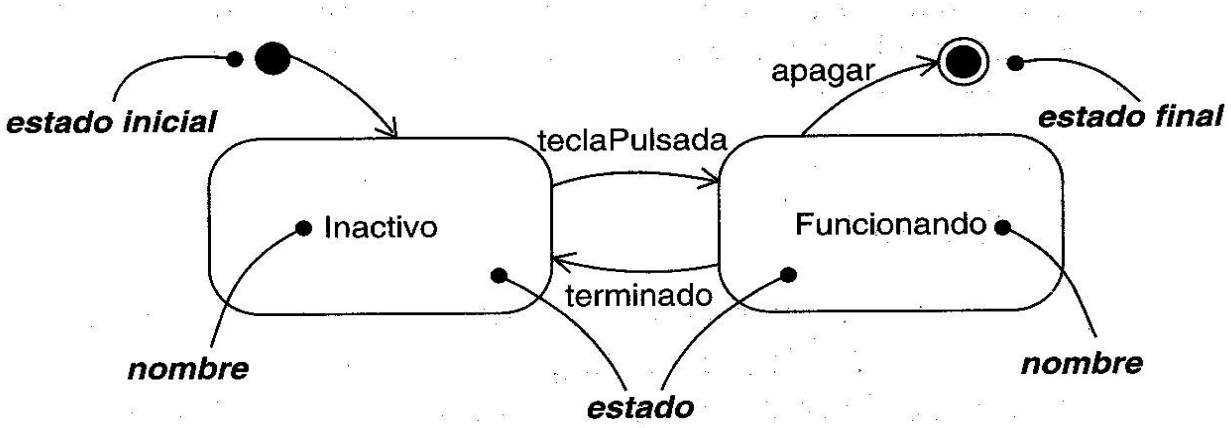
Máquina de Estados - Tipos

- **Ejemplo** de Submáquina de Estados.



Máquina de Estados - Tipos

- **Inicial**: Indica el punto de comienzo por defecto para la máquina de estados o para el subestado.
- **Final**: Indica que la ejecución de la máquina de estados o estado que lo contiene, ha finalizado. Si la máquina tiene uso infinito, puede no tener estado final (pero siempre tendrá estado inicial).



Máquina de Estados - Tipos

- Otros tipos de estados:

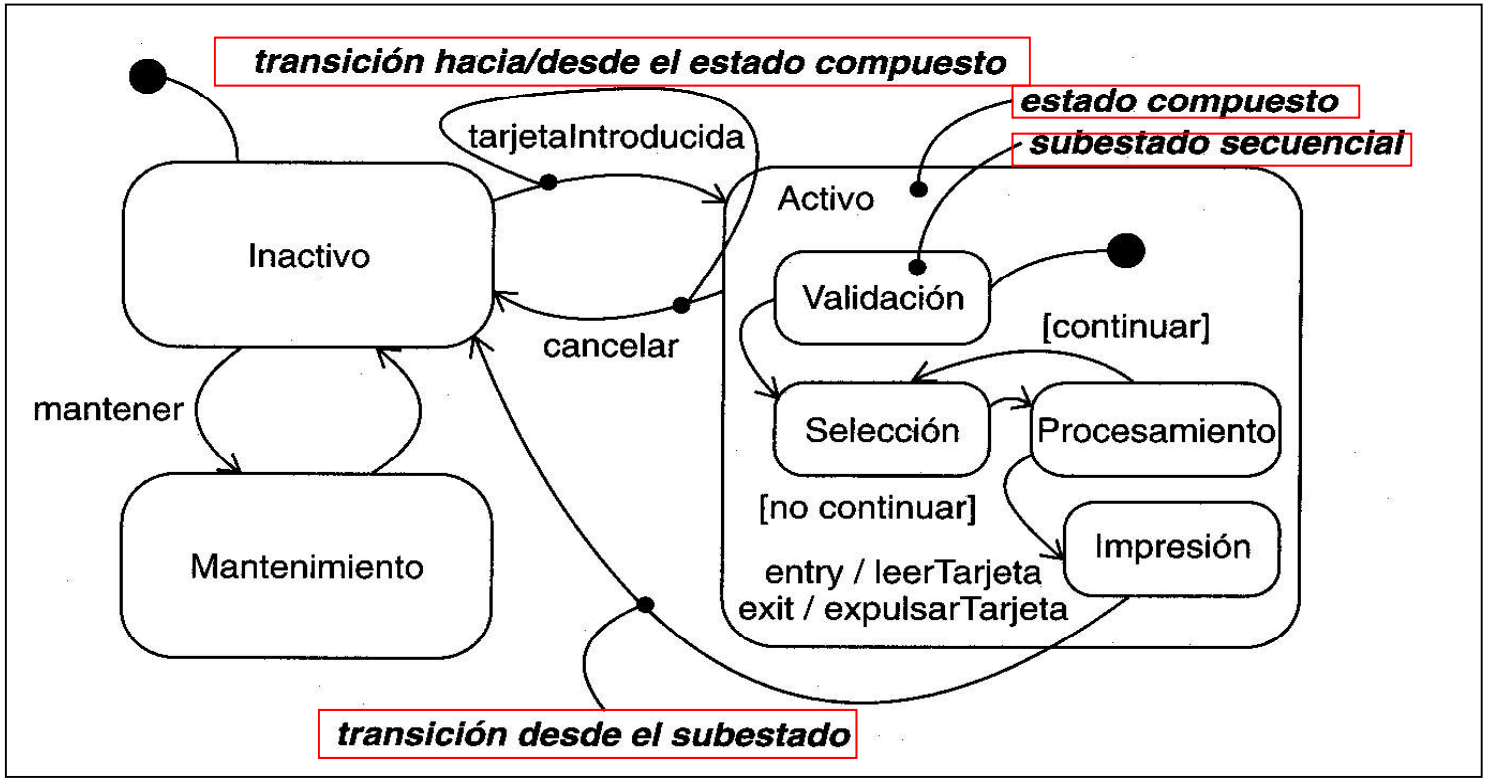
Tipo	Descripción	Notación
finalizador	Un estado especial cuya activación finaliza la ejecución del objeto al que pertenece la máquina de estados	
conjunción	Un pseudoestado que encadena segmentos de transición en una transición de ejecución hasta la finalización	
elección	Un pseudosestado que realiza una bifurcación dinámica dentro de una transición de ejecución hasta la finalización	
punto de entrada	Un pseudoestado externo y visible dentro de la máquina de estados que identifica un estado interno como destino	
punto de salida	Un pseudoestado externo y visible dentro de la máquina de estados que identifica un estado interno como origen	

Máquina de Estados - Compuestos

- Un **subestado** es un estado anidado dentro de un **estado compuesto**.
  - Los subestados dentro de un estado compuesto pueden ser **concurrentes** (estado compuesto ortogonal) o **secuenciales** (estado compuesto no ortogonal).
  - En **UML 2**, un estado compuesto se representa igual que un estado simple, pero con una máquina de estados anidada.
  - Los subestados se pueden anidar a cualquier nivel.
- Una transición desde fuera de un estado compuesto puede apuntar a:
  - El estado compuesto (la máquina de estados anidada debe incluir un estado inicial, al cual pasa el control al entrar al estado compuesto).
  - Un subestado anidado (el control pasa directamente a el).
- Similar ocurre con una transición saliendo de un estado compuesto o de uno de sus subestados anidados.

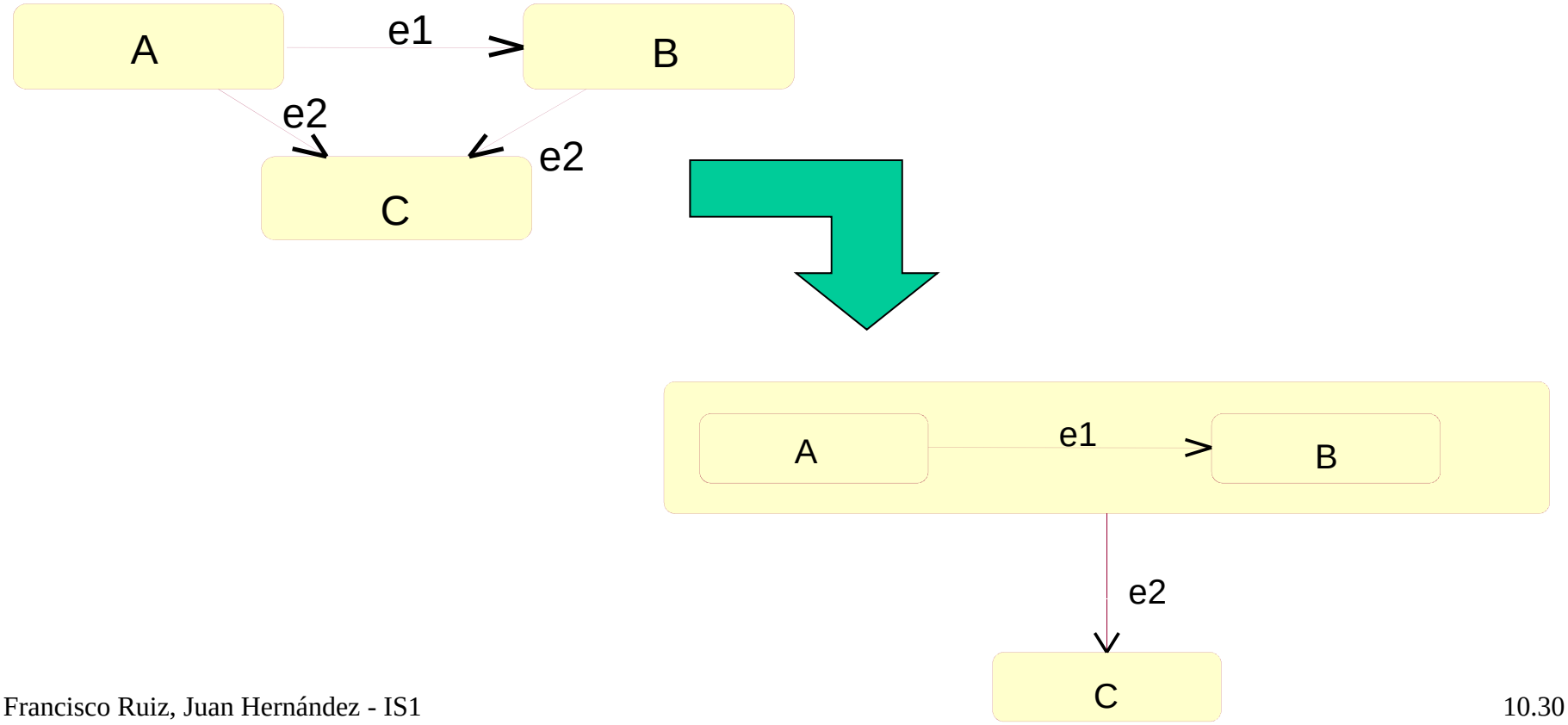
Máquina de Estados - Compuestos

- **Subestados Secuenciales**.
  - **Ejemplo** del Comportamiento de un Cajero Automático.



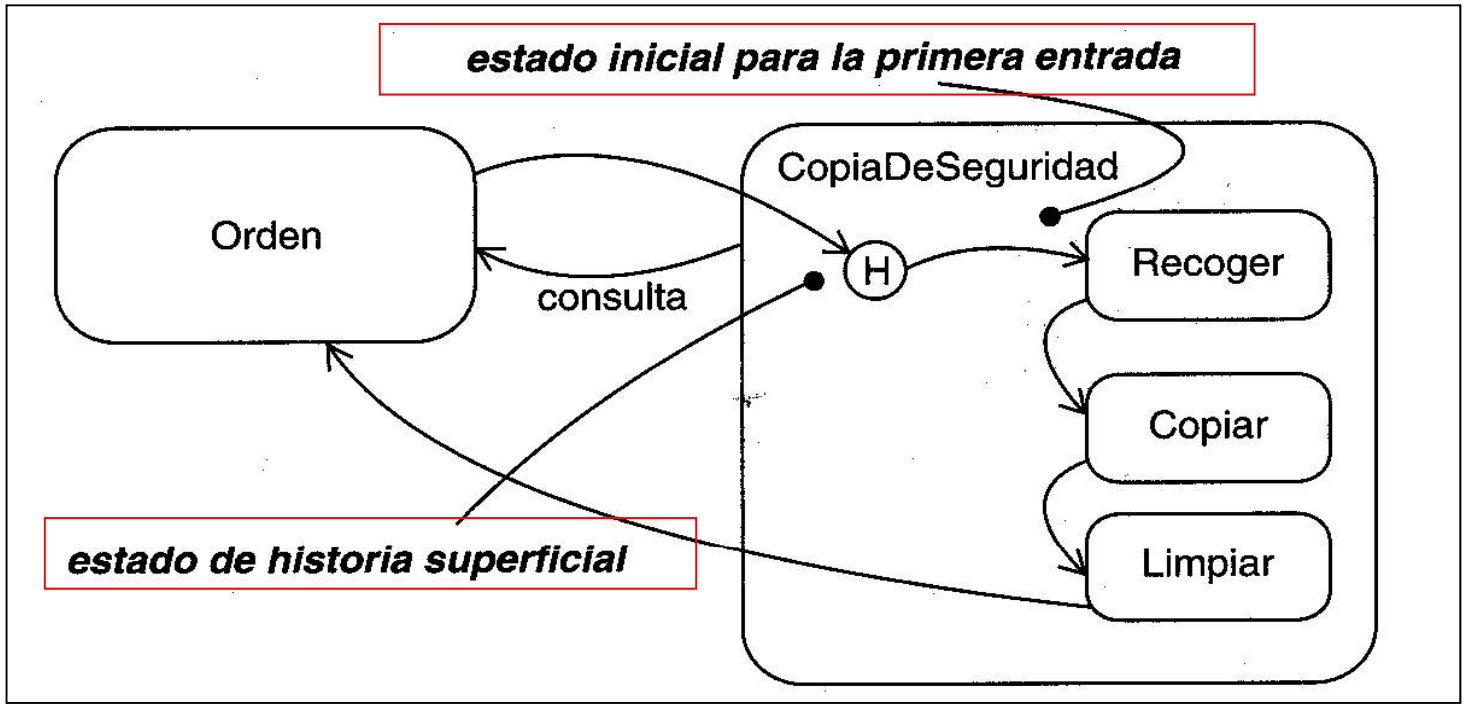
Máquina de Estados - Compuestos

- **Subestados Secuenciales**.
  - Este tipo de estados compuestos es una ayuda para simplificar máquinas de estado mediante un mecanismo de abstracción de agregación de estados dependientes.



Máquina de Estados - Compuestos

- **Estados con Historia**.
  - Un estado compuesto recuerda el último subestado activo antes de la transición que provocó la salida del estado compuesto.

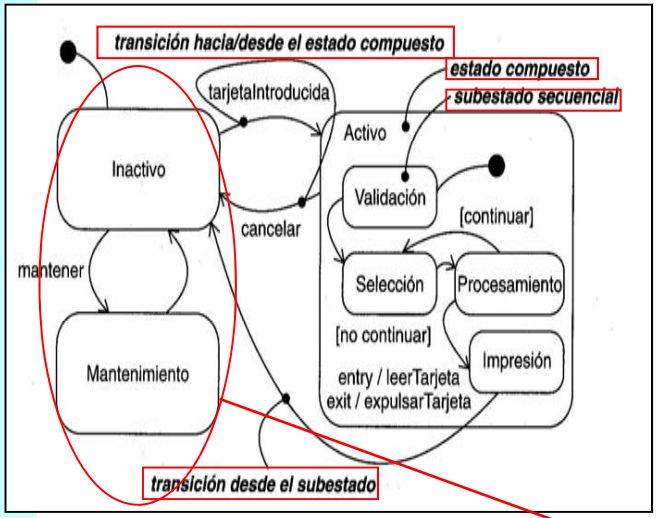


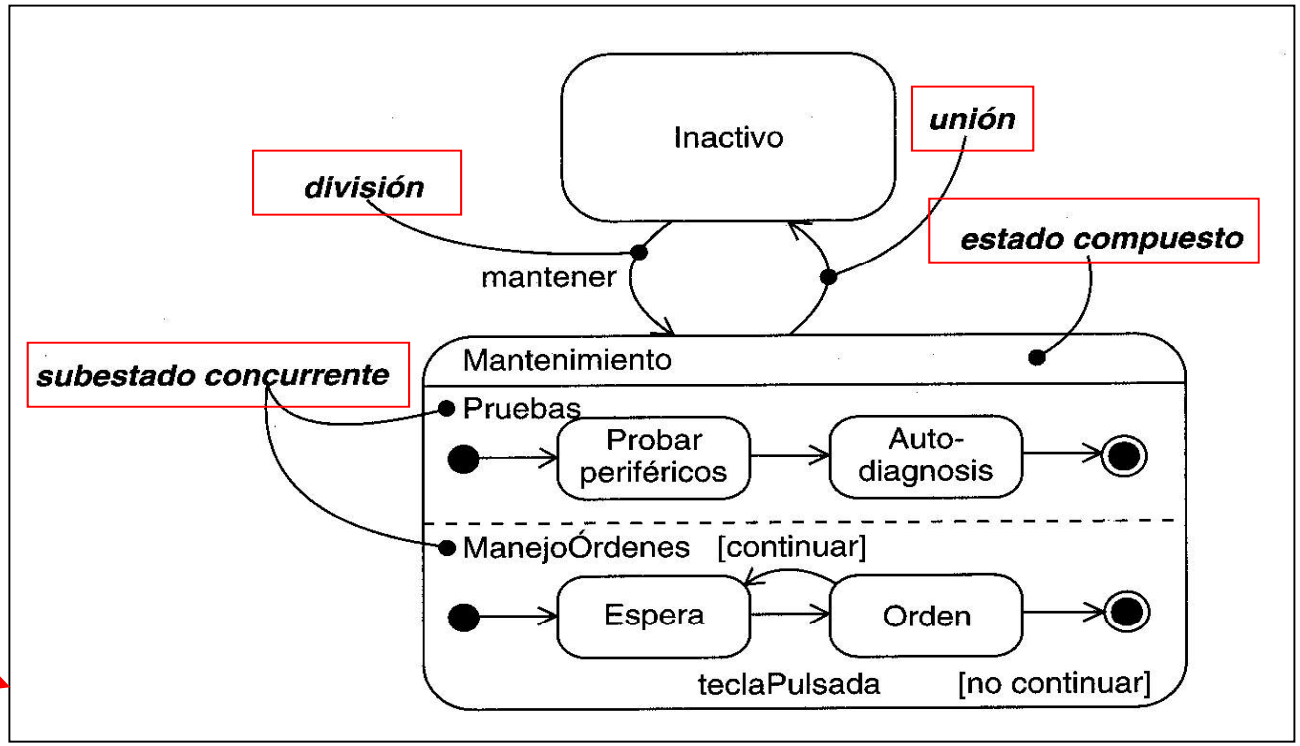
Máquina de Estados - Compuestos

- **Subestados Concurrentes**.
  - Las regiones ortogonales permiten especificar dos o más máquinas de estados anidadas que se ejecutan en paralelo en el contexto del objeto que las contiene.
  - El estado compuesto acaba mediante una sincronización de las regiones ortogonales: las regiones que alcanzan sus estados finales quedan a la espera hasta que todas las regiones acaban, y entonces concluye el estado compuesto.
  - Cada región ortogonal puede tener un estado inicial, un estado final y un estado de historia.

Máquina de Estados - Compuestos

- **Subestados Concurrentes**.
  - **Ejemplo** del Comportamiento de un Cajero Automático.



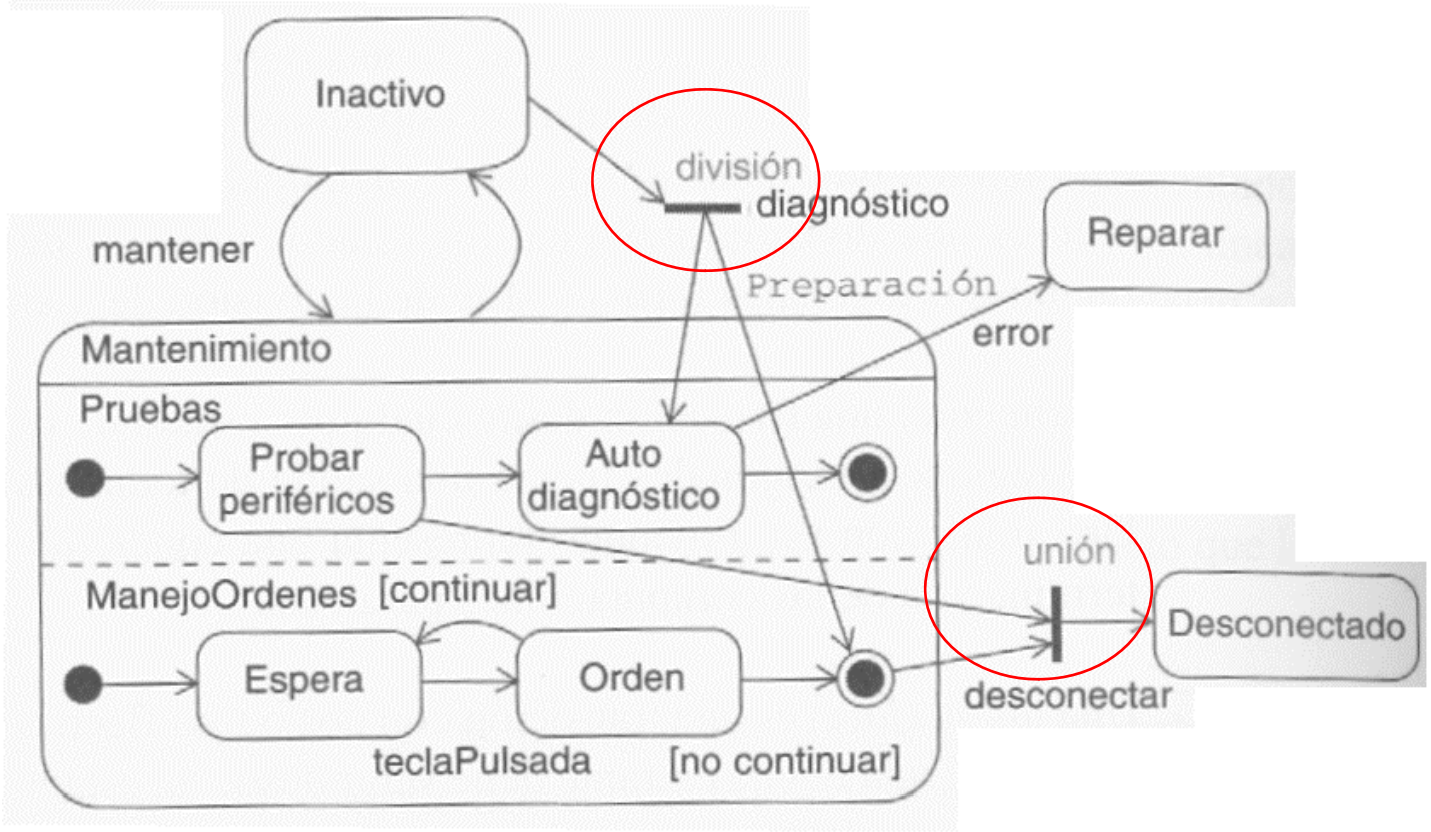


Máquina de Estados - Compuestos

- **Subestados Concurrentes**.
  - Transición de **División** (Fork)
    - El control pasa de un estado simple a varios estados ortogonales, cada uno de una región ortogonal diferente.
    - Las regiones para las que no se especifica subestado destino toman como tal, por defecto, el estado inicial de la región.
  - Transición de **Unión** (Join)
    - Varias entradas, cada una de un subestado de una región ortogonal diferente, pasan el control a un único estado simple.
    - Puede tener un evento disparador.
    - La transición ocurre si todos los subestados origen están activos.
    - El control sale de todas las regiones ortogonales, no solo de las que tienen subestado de entrada a la unión.

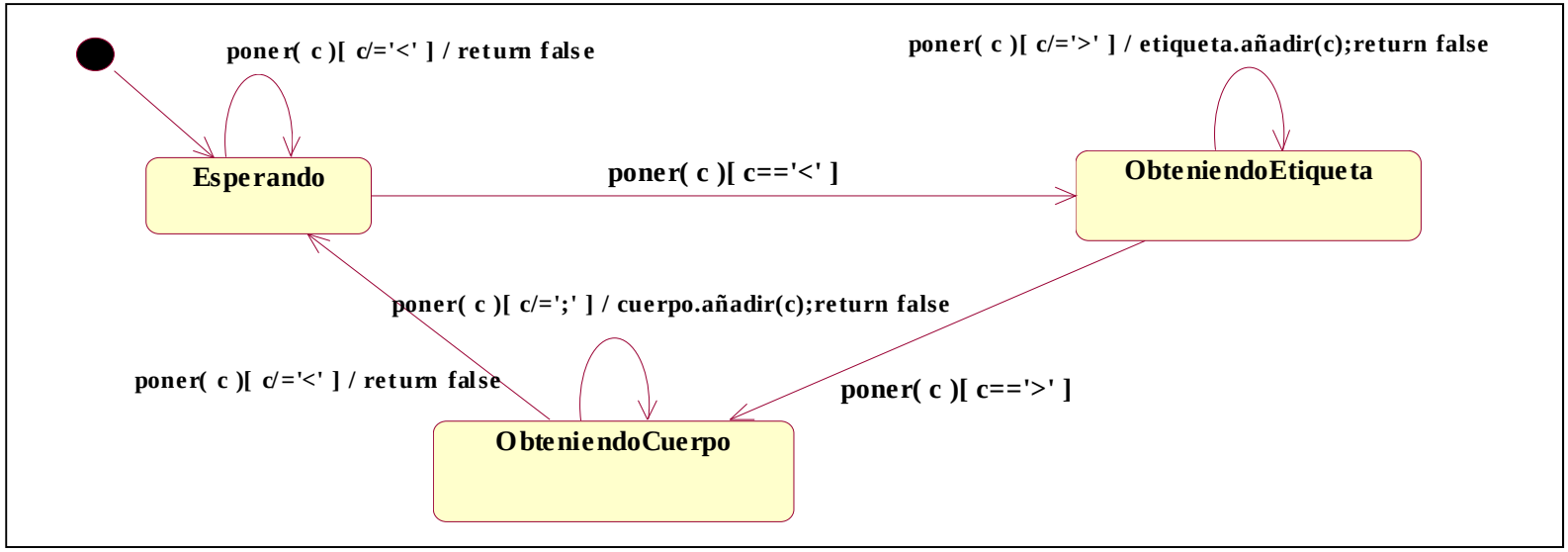
Máquina de Estados - Compuestos

- **Subestados Concurrentes**.
  - Ejemplo con División y Unión.



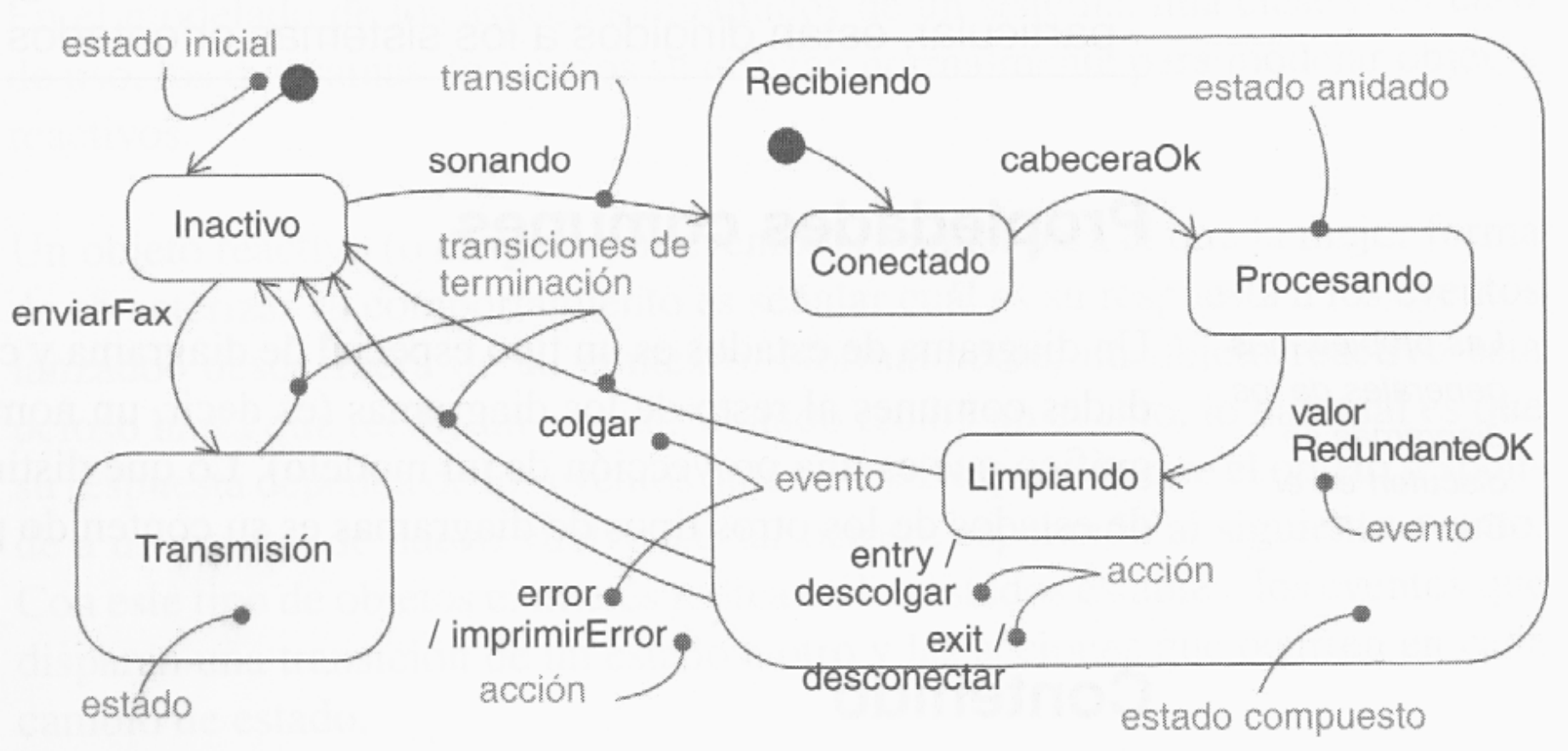
Representación de Diagramas de Estados

- Los **Diagramas de Estados** representan **autómatas de estados finitos**, desde el punto de vista de los estados y las transiciones.
- Gráficamente, en **UML 2** un **Diagrama de Estados** es un grafo dirigido, es decir, una colección de nodos y arcos.
  - Permiten expresar concurrencia, sincronización y jerarquías de objetos.
  - Los estados inicial y final(es) están diferenciados del resto.



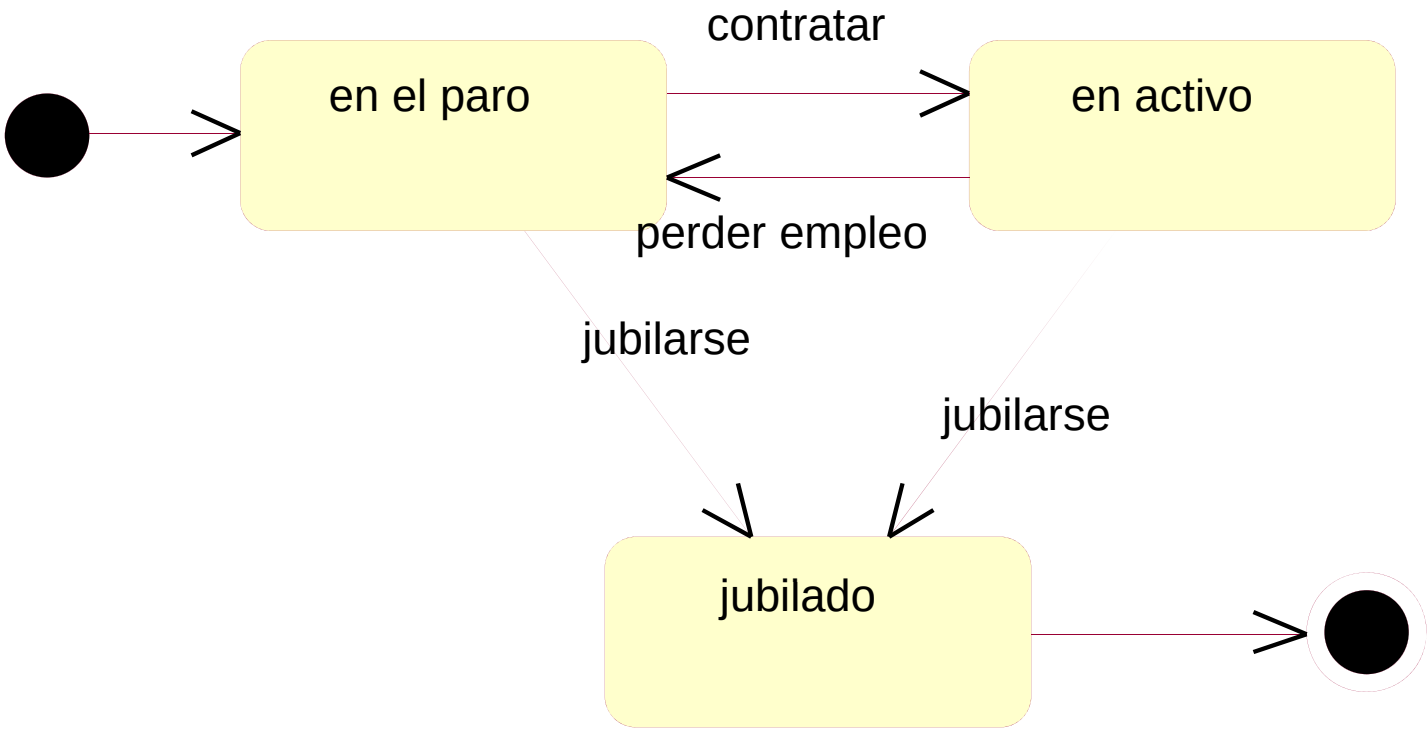
Representación de Diagramas de Estados

- **Ejemplo** de Diagrama de Estados de objetos de una clase.



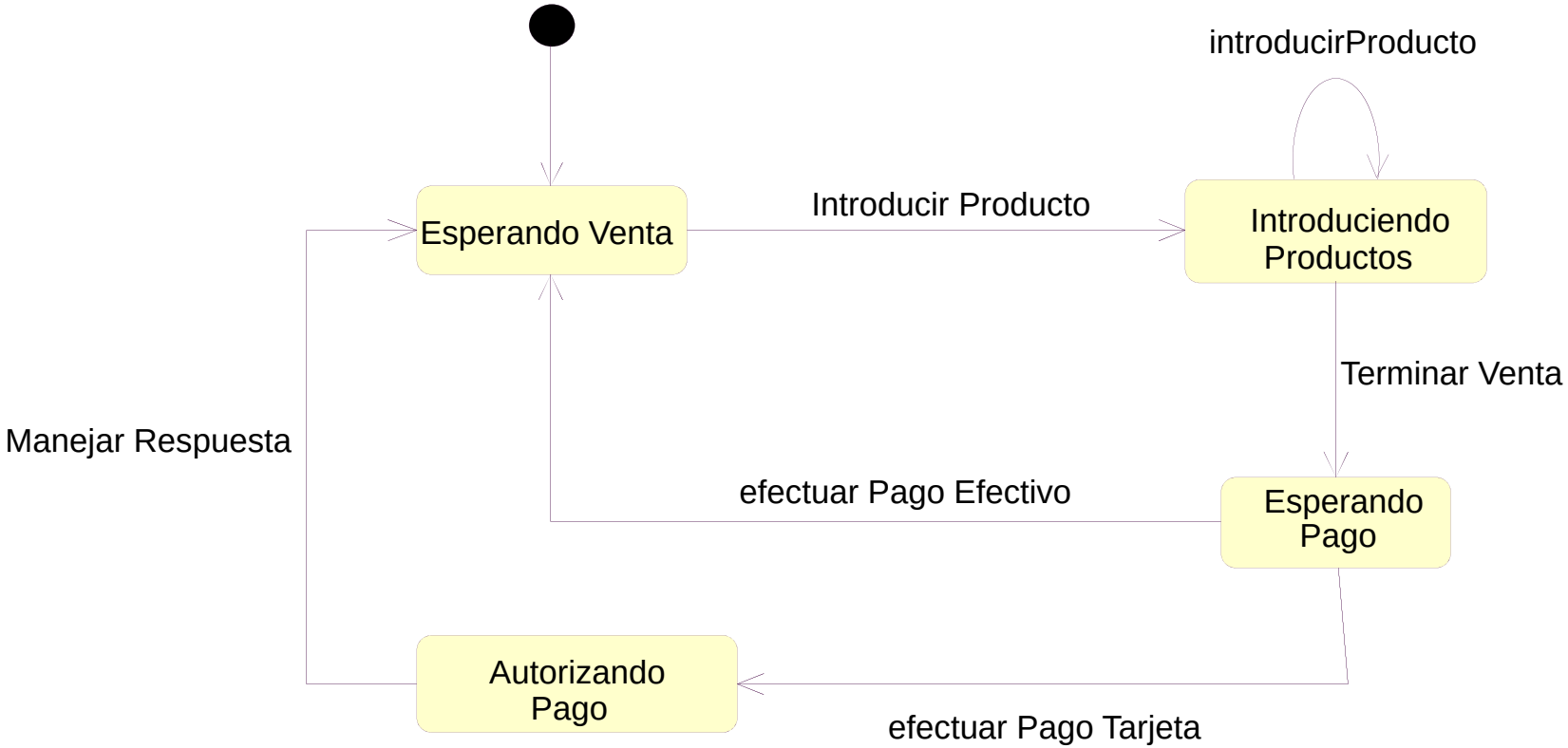
Representación de Diagramas de Estados

- Ejemplo de un Diagrama de Estados para la clase **persona.**



Representación de Diagramas de Estados

- **Ejemplo** de Diagrama de Estados de un caso de uso.



Diagramas de Estado – Consejos

- Un **diagrama de estados** está bien estructurado si:
  - Es **sencillo** (no tiene ningún estado o transición innecesario).
  - Tiene un **contexto claro** (tiene acceso a todos los objetos visibles dentro del objeto que la contiene).
  - Es **eficiente** (su comportamiento lo lleva a cabo con equilibrio entre tiempo y recursos).
  - Es **comprensible** (los nombres de los estados y transiciones provienen del vocabulario del sistema).
  - No tiene muchos **niveles de anidamiento**. (1 o 2 niveles de subestados suele ser suficiente).
  - No abusa de las **regiones ortogonales** (subestados concurrentes).
- Al dibujar **diagramas de estados** en UML 2:
  - Evitar los cruces entre transiciones.
  - Expandir los estados compuestos sólo cuando sea necesario por motivos de comprensión.

Diagramas de Estado – Consejos

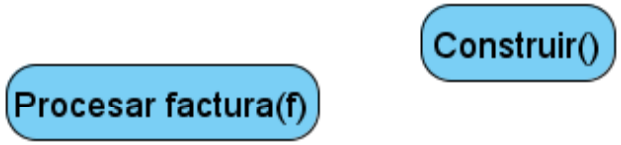
- Al dibujar un **diagrama de estados** en UML 2:
  - Darle un nombre que comunique su propósito.
  - Modelar primero los estados estables del objeto y después, las transiciones legales entre estados.
  - Considerar las bifurcaciones, concurrencia y flujo de objetos como secundarios, pudiendo incluso ir en otros diagramas separados.
  - Organizar los elementos para que se minimicen los cruces de líneas.
  - En los diagramas grandes, considerar el uso de características avanzadas (submáquinas, et.c) para facilitar la comprensión.

Actividades

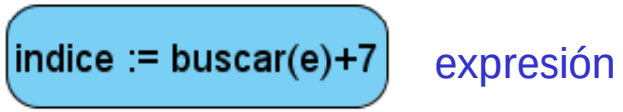
- Una **Actividad** es una ejecución no atómica, dentro de una máquina de estados.
- La estructura de una actividad es un flujo entre **nodos de actividad** (subactividades o acciones).
  - La ejecución de una actividad produce la ejecución de los nodos de actividad incluidos en la actividad.
  - Un nodo de actividad es una agrupación anidada de acciones o de otros nodos de actividad.
  - Una acción es un tipo de nodo de actividad atómico (que no puede descomponerse).
  - Nodos de actividad y acciones se representan igual.

Actividades

- Los nodos de actividad sirven como unidades organizativas dentro de las actividades.



- Las acciones son computaciones ejecutables atómicas, como:
  - Llamadas a otras operaciones,
  - Envío de señales,
  - Creación o destrucción de objetos, o
  - Simples cálculos
    (evaluación de una expresión) ...

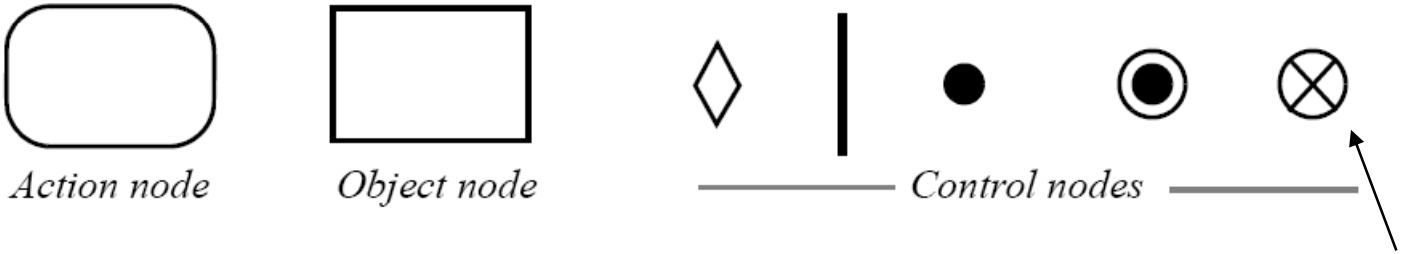


Diagramas de Actividades

- Un **Diagrama de Actividades** muestra el flujo de control entre actividades.
- Cada diagrama representa una actividad, que puede estar formada por otras actividades más pequeñas.
- Mientras un diagrama de interacción muestra objetos que pasan mensajes, uno de actividades muestra las operaciones que se pasan entre objetos.
- Sirven para modelar:
  - La dinámica de un conjunto de objetos.
  - El flujo de control de una operación o un caso de uso.
  - Un proceso de negocio o un flujo de trabajo (Workflow).

Diagramas de Actividades - Contenido

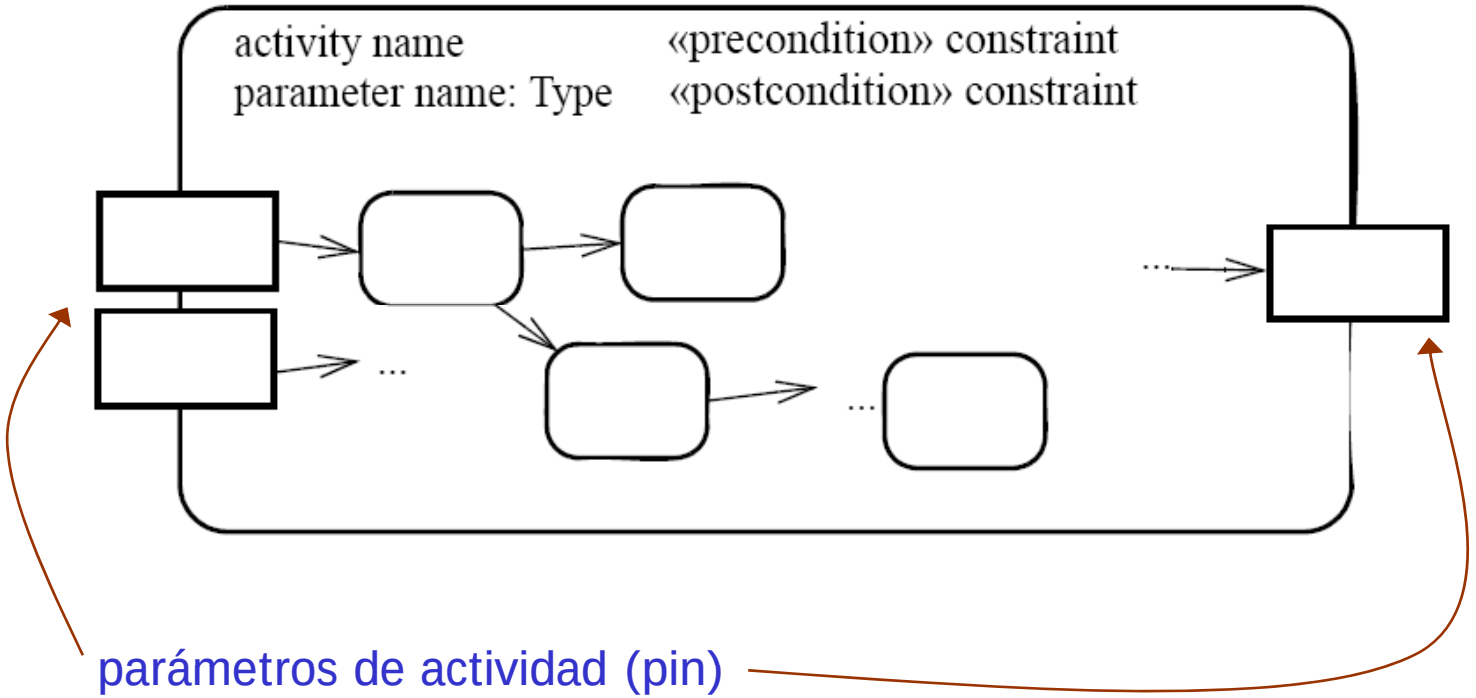
- Normalmente, los **diagramas de actividades** contienen:
  - **Nodos**
    - **Acciones** (nodos de actividad atómicos).
    - **Actividades** (nodos de actividad con estructura interna).
    - **Nodos de Control** (controlan el flujo).
    - **Objetos** de valor (objetos o datos utilizados).



  - **Flujos**.
    - **Flujos de Control**.
    - **Flujos de Objetos**.
- También pueden contener notas y restricciones.

Diagramas de Actividades - Contenido

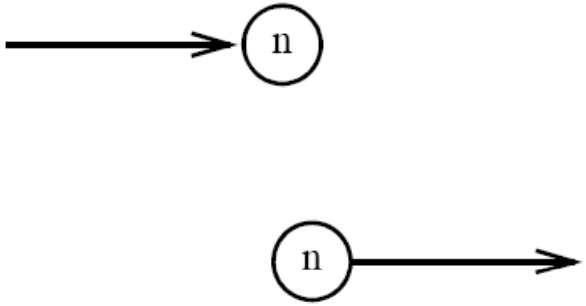
- En **UML 2**, una **actividad** se representa encerrando los nodos y flujos que incluye por un rectángulo de puntas redondeadas, indicando el nombre en la esquina superior-izquierda.

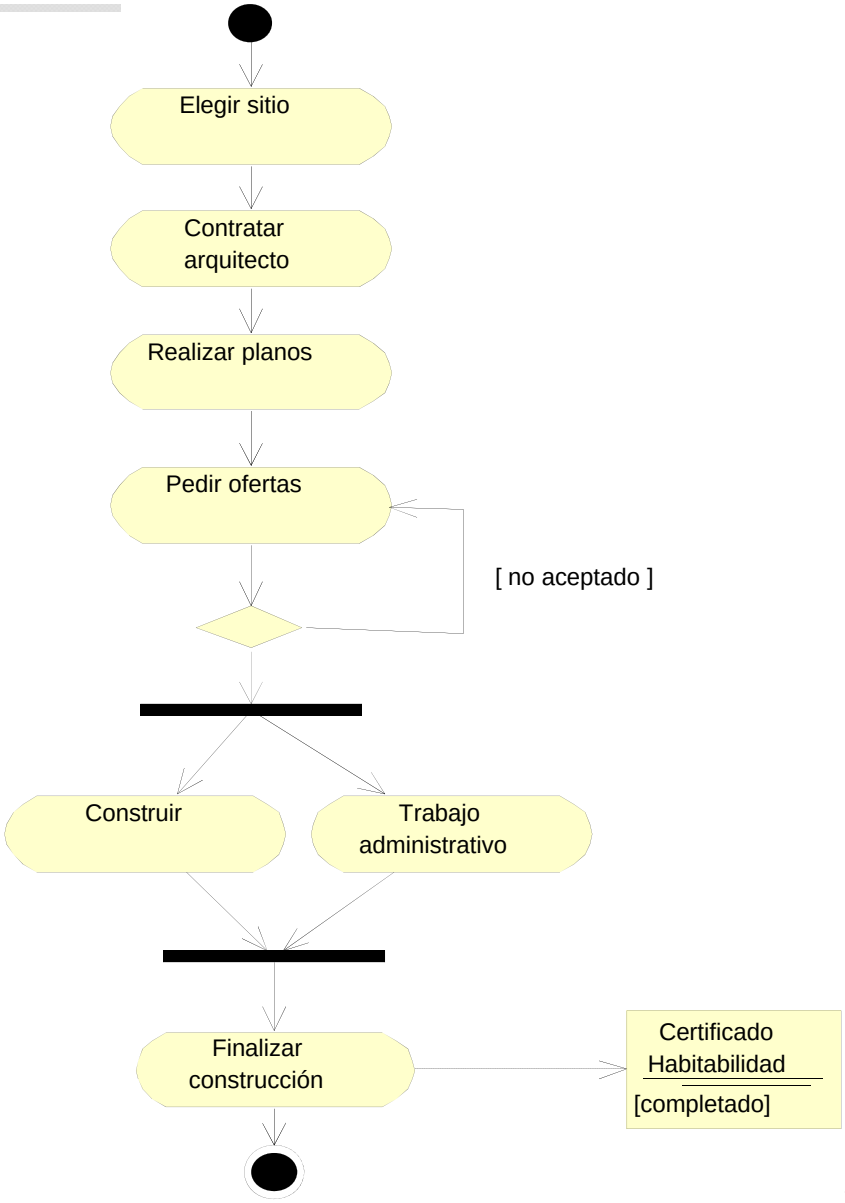


Diagramas de Actividades - Contenido

- **Ejemplo** de diagrama de actividades.

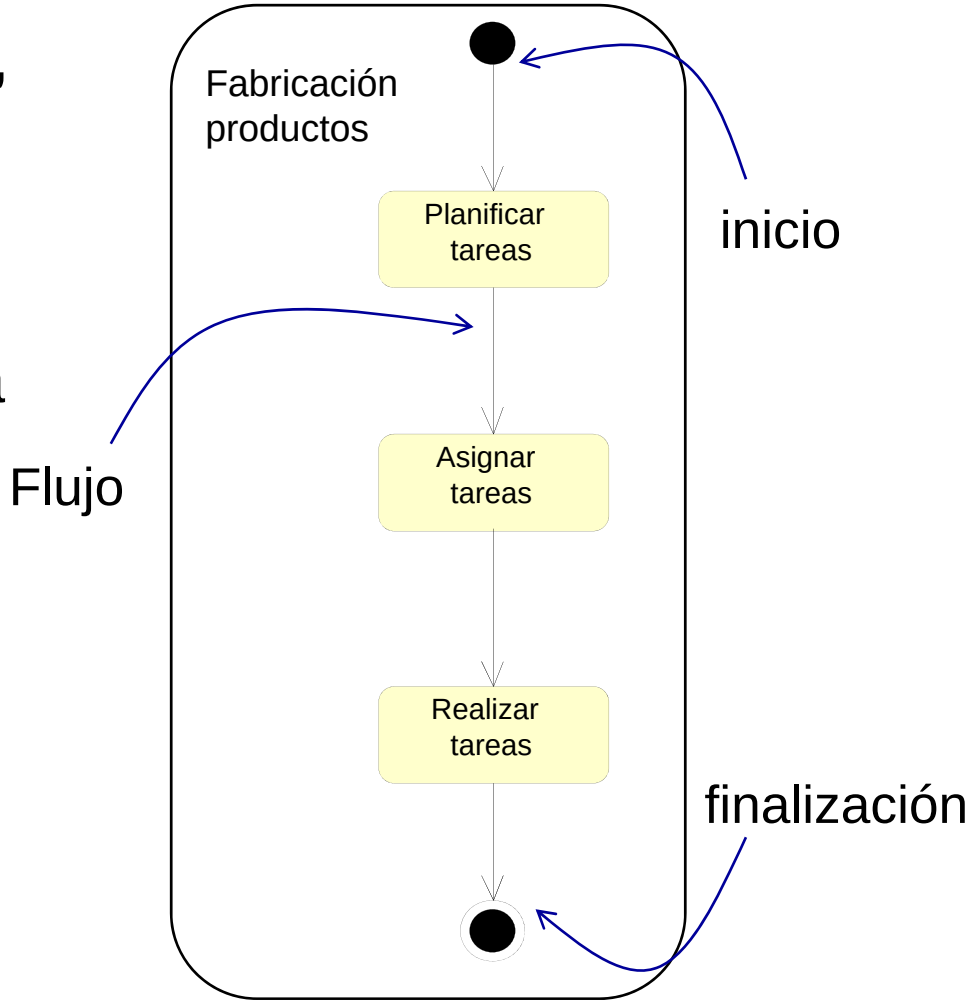
- Se pueden usar conectores.





Diagramas de Actividades – Flujo de Control

- Cuando se completa una acción o un nodo de actividad, el flujo de control pasa a la siguiente acción o nodo de actividad.
- En UML 2 el flujo se especifica con una flecha desde el nodo/acción predecesor al sucesor.
- Debe haber un inicio (evento de inicio) y una terminación (eventos de finalización, pueden ser varios).

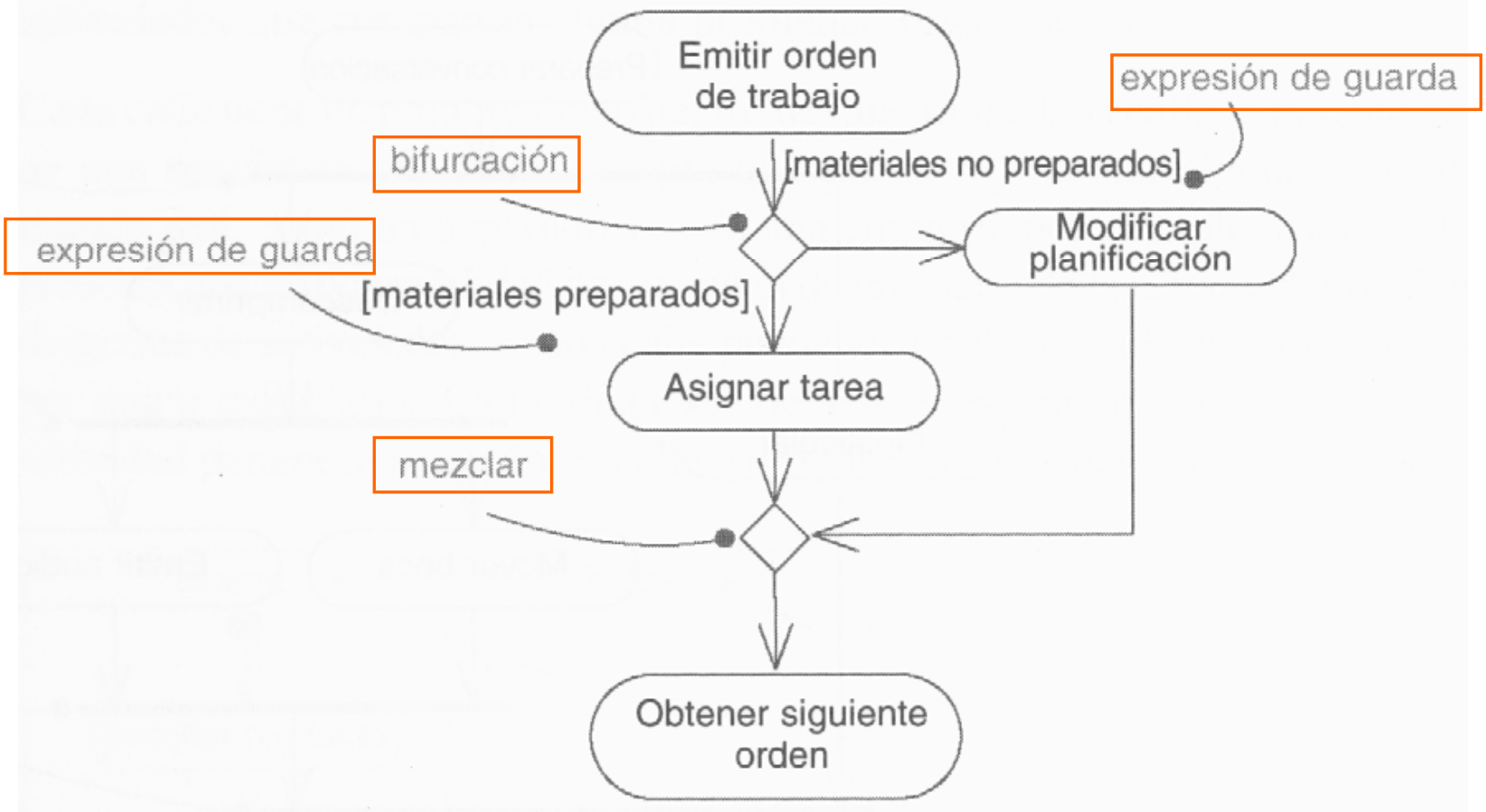


Diagramas de Actividades – Flujo de Control

- Existen varios tipos de nodos de control:
- **Bifurcaciones**. [Decision Node]
  - Representan caminos alternativos, elegidos en función del valor de una expresión booleana.
  - Se representa con un **rombo**.
  - Puede tener un flujo de entrada y dos o más de salida.
  - En cada flujo de salida se coloca una guarda (expresión booleana), que se evalúa al entrar en la bifurcación.
    - Las guardas no deben solaparse (para que solo una sea cierta a la vez).
    - Pero deben cubrir todas las posibilidades (para que siempre haya una cierta).
    - Se puede usar "else" para marcar un flujo de salida alternativo.
- **Fusiones**. [Merge Node]
  - Los caminos antes separados se pueden volver a juntar en un rombo con varias entradas y una salida. Aquí no hay guardas.

Diagramas de Actividades – Flujo de Control

- Ejemplo con **bifurcaciones y fusiones**.

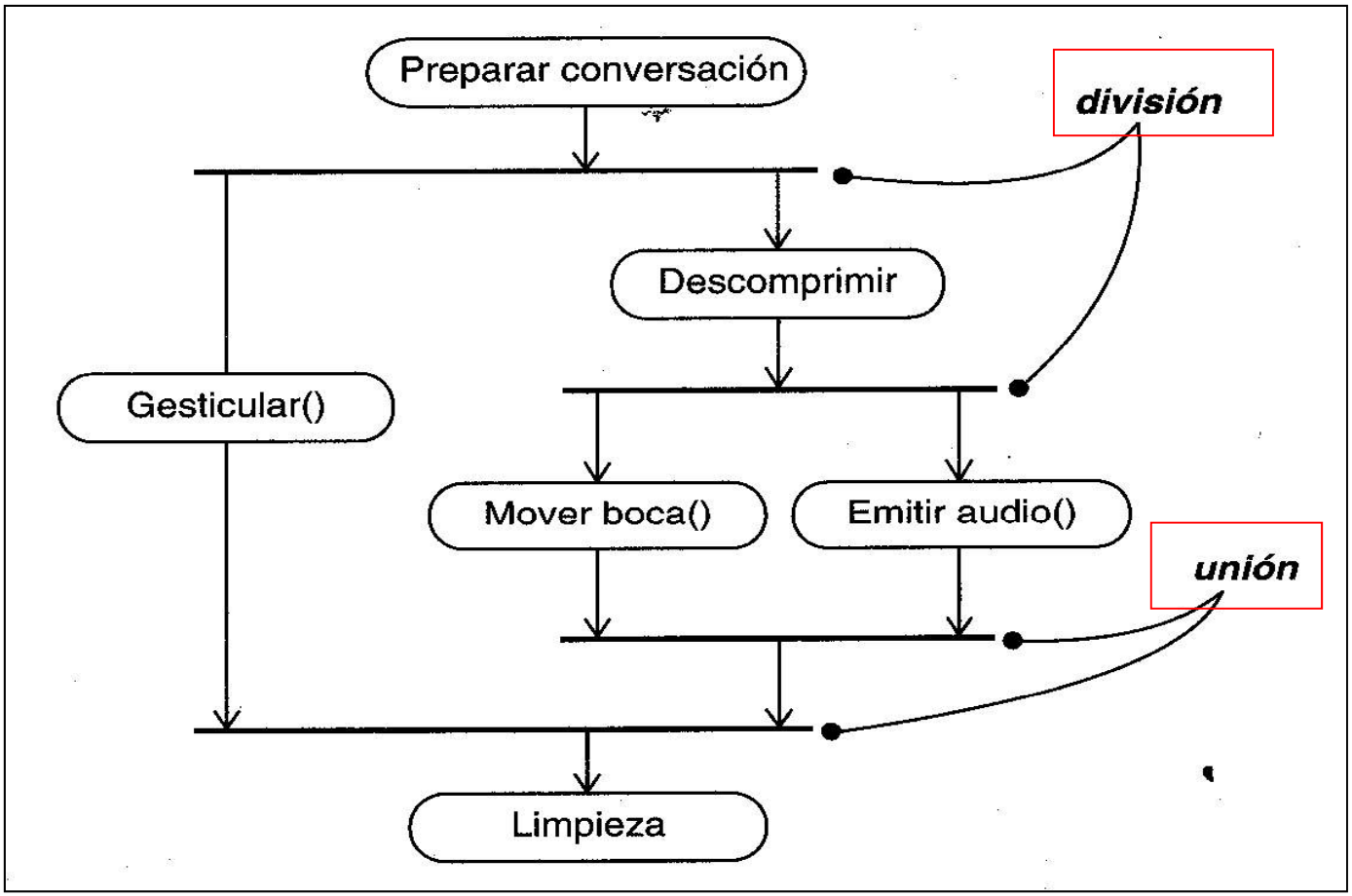


Diagramas de Actividades – Flujo de Control

- **Divisiones**. [Fork Node]
  - Representan la separación de un flujo de control sencillo en dos o más **flujos de control concurrentes**.
  - Tienen una transición de entrada y dos o más de salida., cada una de las cuales representa un flujo independiente.
  - Las actividades de cada camino después de la división continúan en paralelo.
- **Uniones**. [Join Node]
  - Representan la sincronización de dos o más flujos de control concurrentes.
    - Cada flujo de entrada espera hasta que todos han alcanzado la unión.
  - Tienen dos o más transiciones de entrada y una de salida.
- Uniones y Divisiones deben estar equilibradas.

Diagramas de Actividades – Flujo de Control

- Ejemplo con **divisiones y uniones**.



Diagramas de Actividades – Flujo de Control

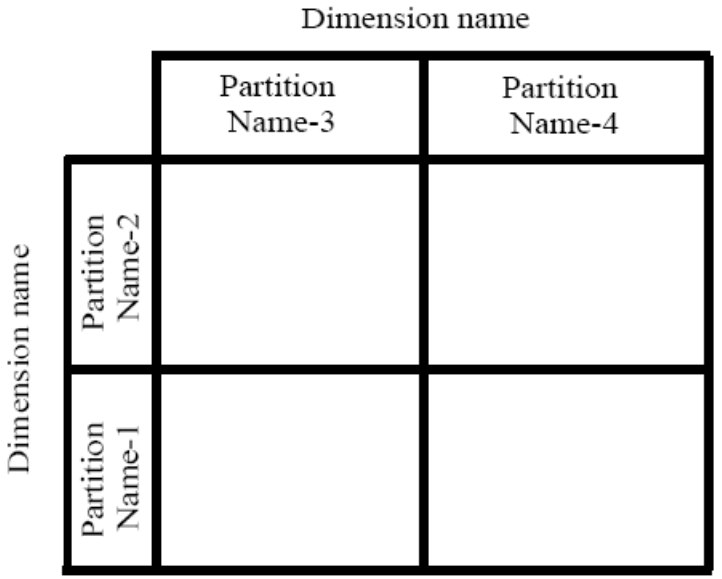
- **Particiones**. [Partition / Swimlane]
  - Son agrupaciones de flujos de procesos.
  - Tienen un nombre único dentro del diagrama.
  - Las transiciones pueden cruzarlas.
  - Cada partición representa una responsabilidad de alto nivel de la actividad global de un diagrama de actividades.
    - Al modelar procesos de negocio cada partición representa una unidad organizacional o una organización diferente, responsable de la realización de la parte correspondiente.
  - Conceptualmente, las actividades de cada partición se consideran (casi siempre) independientes de las actividades de las demás particiones.

Diagramas de Actividades – Flujo de Control

- **Particiones**.
  - En **UML 2** pueden ser unidimensionales simples (**calles**) o estructuradas, o bidimensionales.

a) Partition using a swimlane notation

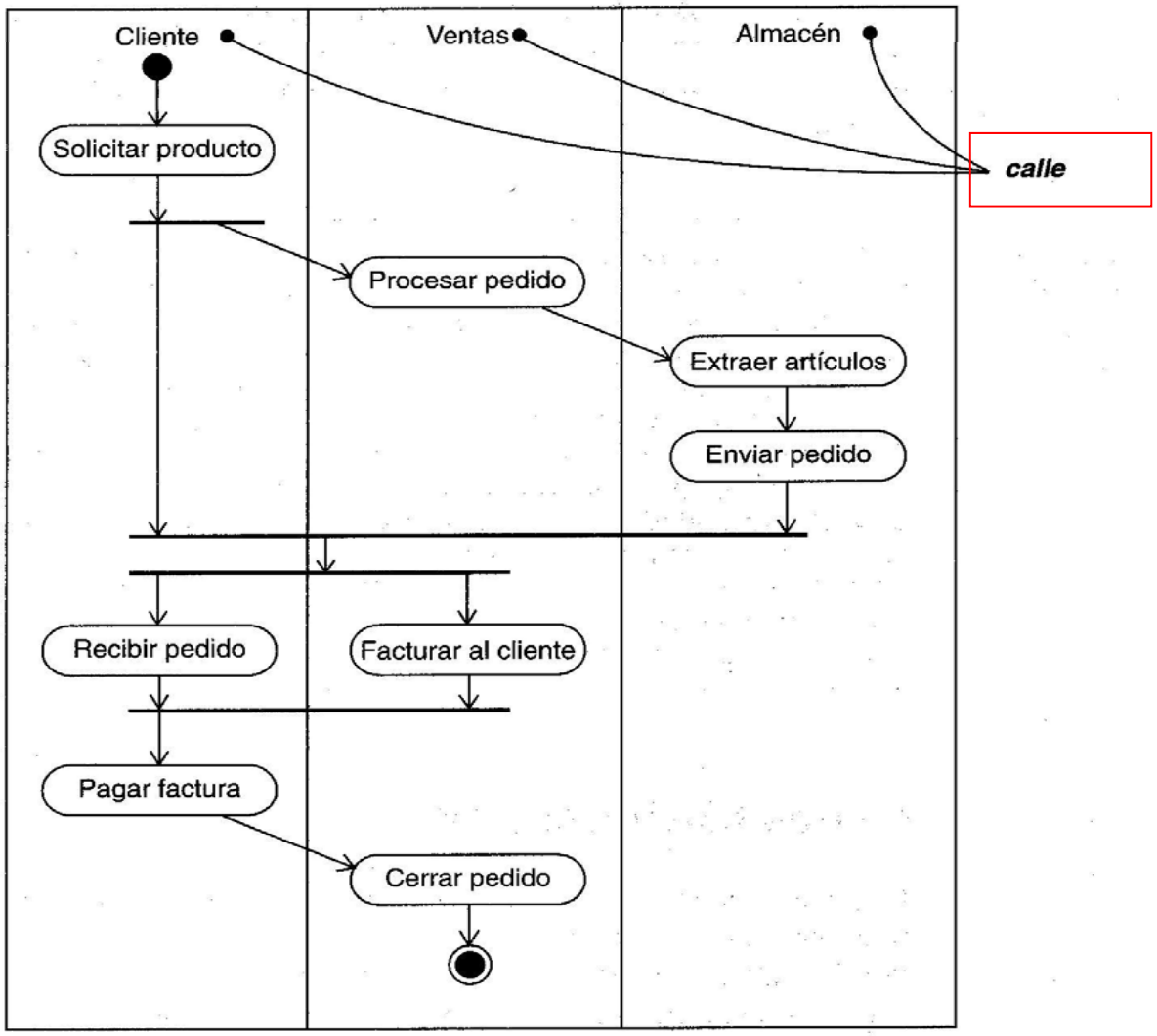
b) Partition using a hierarchical swimlane notation



c) Partition using a multidimensional hierarchical swimlane notation

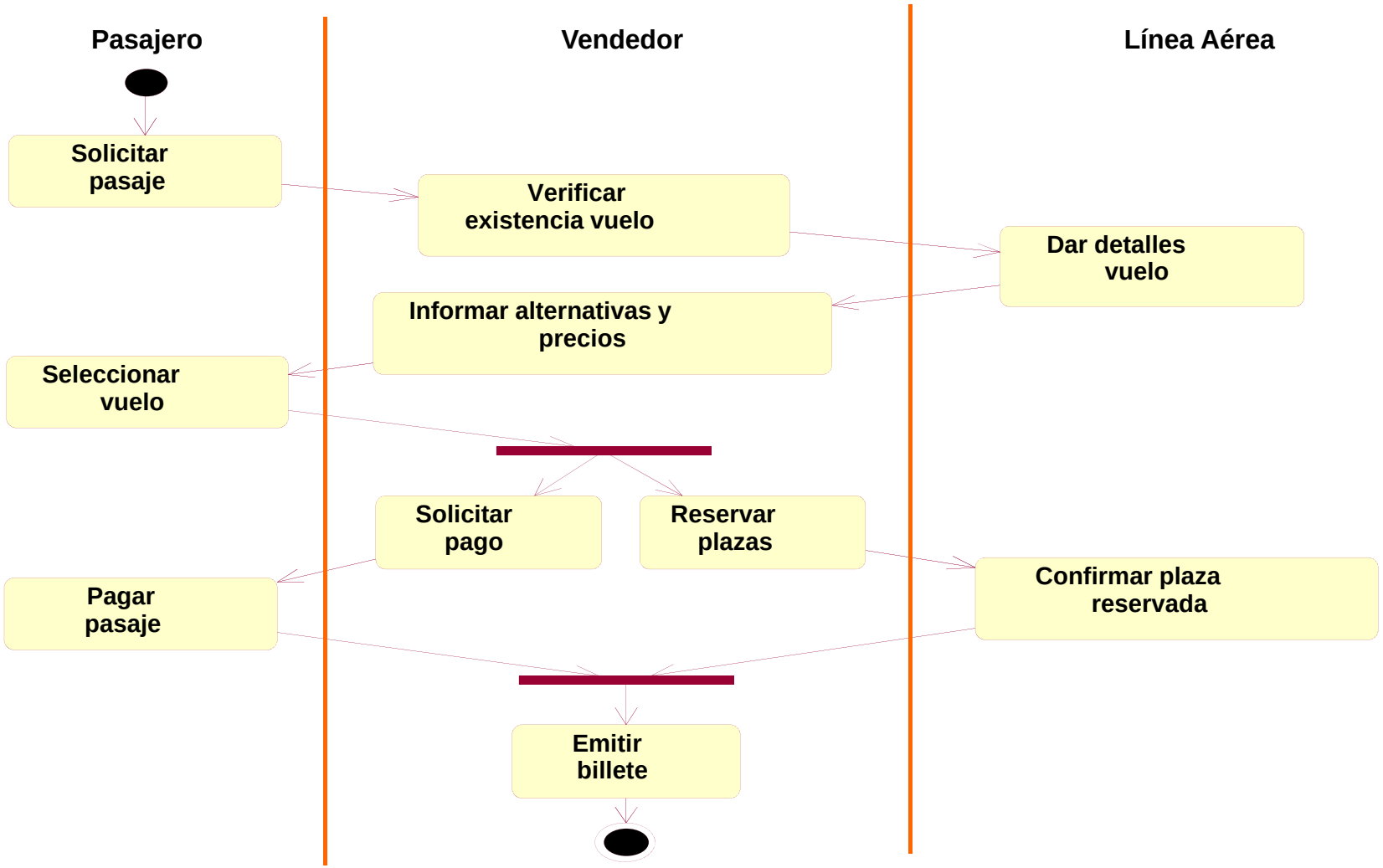
Diagramas de Actividades – Flujo de Control

- Ejemplo con **particiones**. Gestión de Pedidos.



Diagramas de Actividades – Flujo de Control

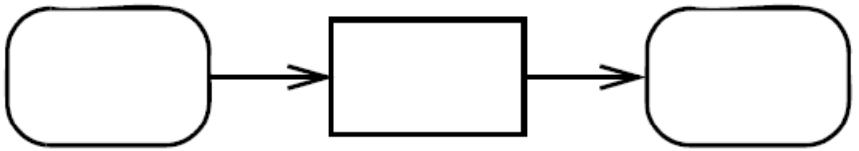
- Ejemplo con **particiones**. Venta de billetes de avión.



Diagramas de Actividades – Flujo de Objetos

- Los **Flujos de Objetos** son flujos en los cuales se ven involucrados objetos.
  - Los objetos se representan como nodos objeto conectados con flechas a las acciones que los crean o los consumen.

Object flow
(without activity nodes)

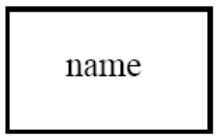


Two object flow edges linking object nodes and actions

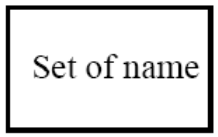
  - También se puede mostrar cómo cambia el estado del objeto.
    - Se muestra el estado entre corchetes debajo del nombre del objeto.

Diagramas de Actividades – Flujo de Objetos

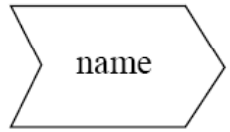
- **Nodos Objeto**.
  - Indican que una instancia de un clasificador (opcionalmente en un cierto estado) está disponible en un punto particular de una actividad.



Object node

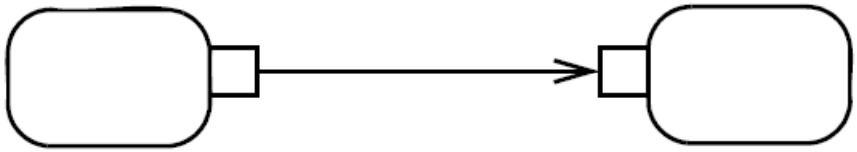


Object node for tokens containing sets



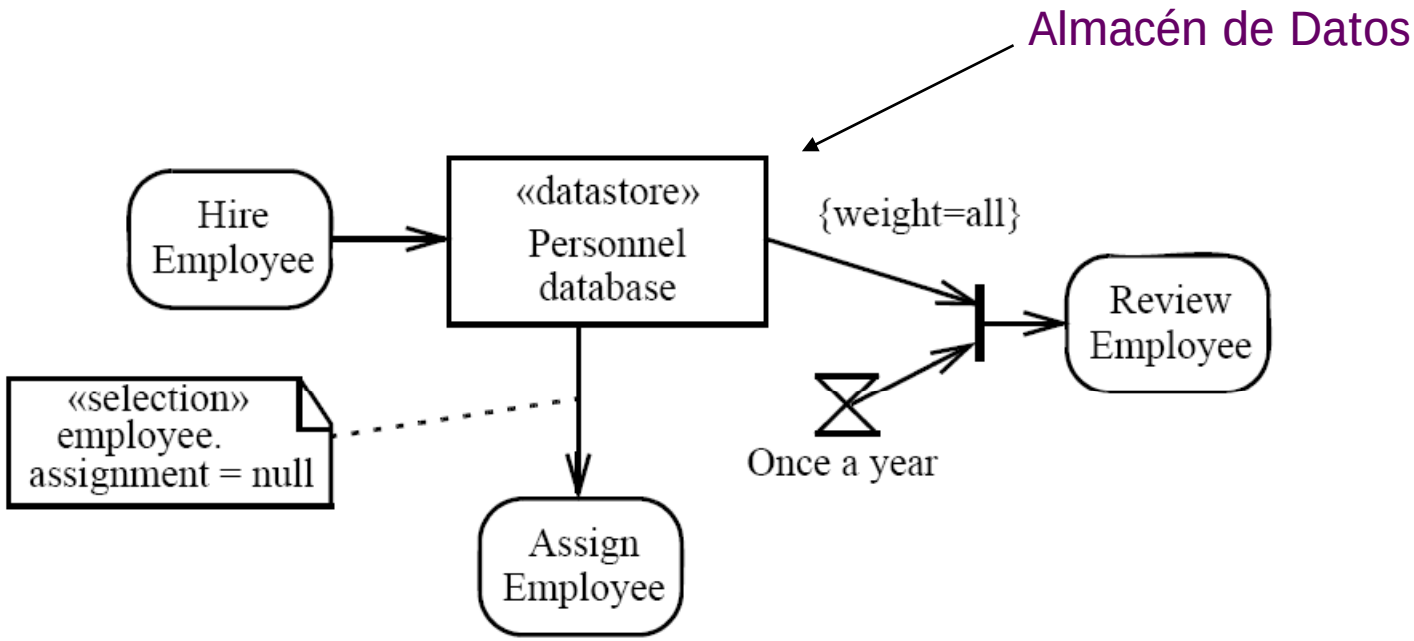
Object node for tokens with signal as type

- **Parámetros de Acciones**. [Pin]
  - Son una manera especial de flujo de objetos.
  - Representan elementos que proveen valores de entrada para las acciones o valores resultantes de ellas.
  - Se representan como pequeños cuadrados en el borde del símbolo de la acción.



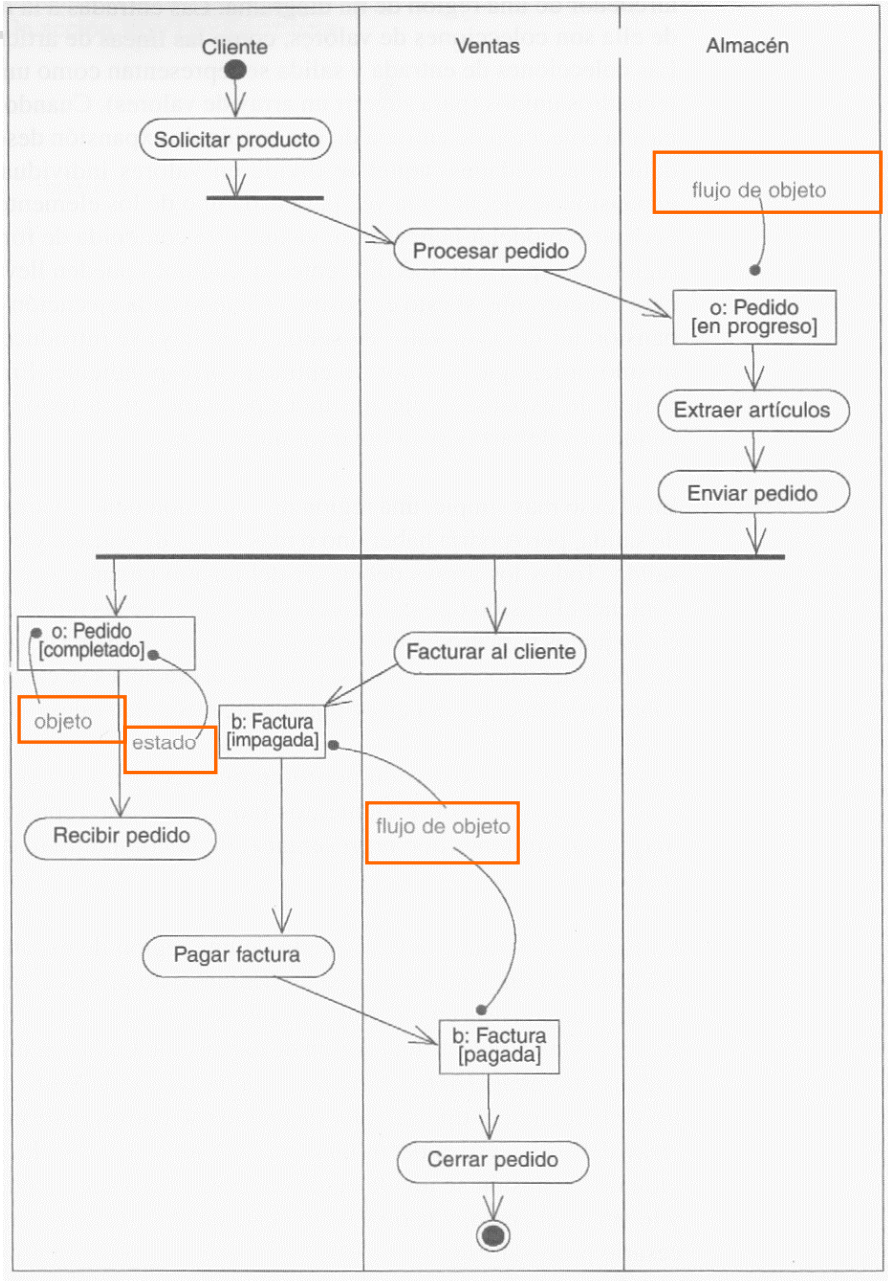
Diagramas de Actividades – Flujo de Objetos

- **Almacen de Datos**. [Data Store]
  - Tipo especial de nodo objeto que representa un repositorio persistente de información.



Diagramas de Actividades – Flujo de Objetos

- Ejemplo con **flujos de objetos**.

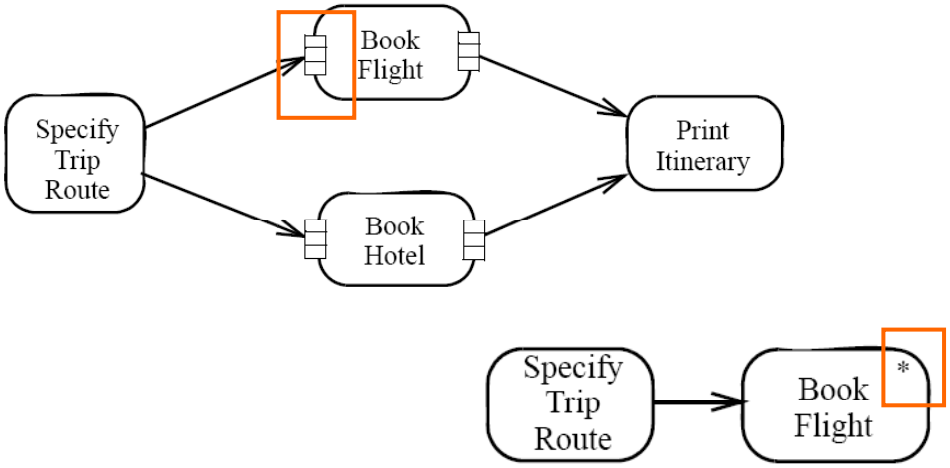


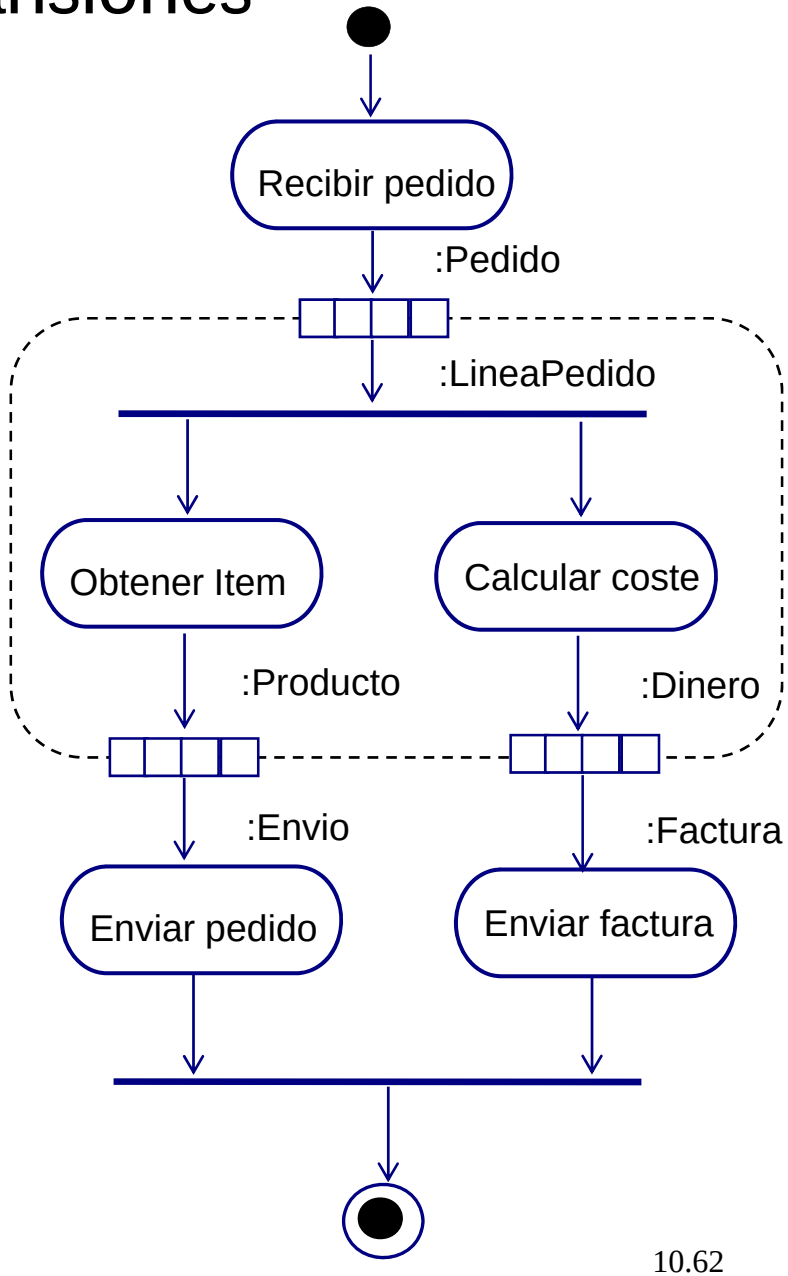
Diagramas de Actividades – Expansiones

- Una **Región de Expansión** representa un fragmento de diagrama de actividades que se ejecuta para todos los elementos de una lista o conjunto.
  - Lleva a cabo un **FOR ALL** sobre los elementos de una colección de valores de entrada para crear una nueva colección de salida (o varias).
  - La iteración está implícita => no es necesario modelarla.
  - Hay tres maneras de realizar las ejecuciones para los valores de entrada:
    - **«parallel»**: ejecuciones independientes concurrentes.
    - **«iterative»**: Ejecuciones dependientes quew se deben ejecutar una detra´s de otra en el mismo orden que los valores (opción por defecto).
    - **«stream»**: Una única ejecución para una lista de valores de entrada.
  - Todas las colecciones de entrada y de salida tienen el mismo número de elementos, aunque pueden ser de distinto tipo.
  - Puede haber una o varias colecciones de entrada y cero o más de salida.

Diagramas de Actividades – Expansiones

- En **UML 2**, una **Región de Expansión** se representa con una línea discontinua alrededor.
  - Las entradas y salidas a la región se representan como una fila de pequeños recuadros unidos (imitando un vector).
- También hay versiones reducidas de la notación:





Diagramas de Actividades – Consejos

- Un **Diagrama de Actividades** bien estructurado:
  - Modela un aspecto de la dinámica de un sistema.
  - Contiene sólo aquellos elementos necesarios para comprender ese aspecto.
  - Proporciona detalles de forma consistente con su nivel de abstracción.
    - Sólo muestra los adornos necesarios para su comprensión.
  - No olvida algún aspecto importante para entender su semántica.

- Evitar diagramas demasiado complejos. Es preferible utilizar varios diagramas de actividades para modelar la dinámica de un sistema.

Diagramas de Actividades – Consejos

- Al dibujar un **Diagrama de Actividades**:
  - Darle un nombre que comunique su propósito.
  - Modelar primero el flujo principal, dejando para después las bifurcaciones, concurrencia y los flujos de objetos.
    - En caso de necesidad por razones de sencillez, pasar dichos aspectos a otro(s) diagrama(s).
  - Situar los elementos para minimizar los cruces de líneas.
  - Usar notas y colores como señales visuales para llamar la atención sobre las características importantes del diagrama.

Apéndice A: Modelado

- Ambos tipos de diagramas permiten modelar aspectos de comportamiento dinámicos, en el contexto del sistema global, un subsistema o una clase; pero ..

- Diagramas de Estados
  - Los aspectos dinámicos pueden involucrar el **comportamiento dirigido por eventos** de cualquier tipo de objeto.
  - Preferibles para modelar **objetos reactivos**.
- **vs**
- Diagramas de Actividades
  - Los aspectos dinámicos pueden involucrar la **actividad** de diversos clasificadores en cualquier parte del sistema.
  - Preferibles para modelar un **flujo de trabajo** o una **operación**.
    - Se adaptan mejor al modelado del flujo de actividades a lo largo del tiempo (tipo diagrama de flujo).

Apéndice A: Modelado – Flujo de Trabajo

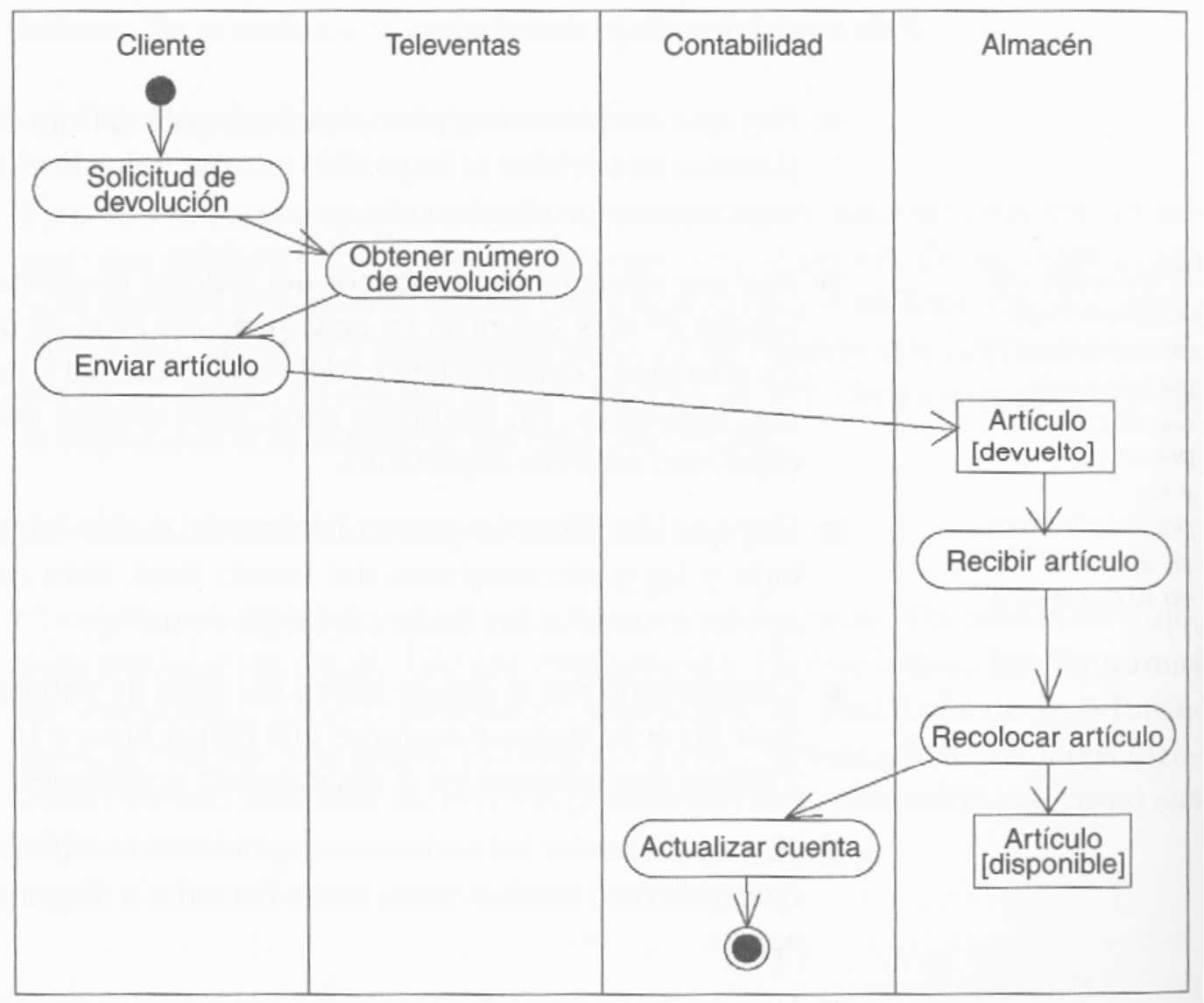
- Los **diagramas de actividades** pueden utilizarse para modelar los **flujos de trabajo** de alto nivel (**procesos de negocio**) en los que colaboran sistemas automáticos y actores humanos.
  - Existe otro lenguaje estándar específico para procesos de negocio (BPMN).
- Para modelar un Flujo de Trabajo (FT):
  1. Identificar el aspecto de interés para el FT.
  2. Seleccionar los objetos del negocio que tienen las responsabilidades de más alto nivel en cada parte del FT.
     - Pueden ser vocabulario del sistema o más abstractos.
     - Crear una calle para cada actor de negocio importante.
  3. Identificar las precondiciones del estado inicial del FT y las post-condiciones del estado final (modelar los límites del FT).

Apéndice A: Modelado – Flujo de Trabajo

- Para modelar un Flujo de Trabajo (FT): (cont.)
  4. Empezando por el estado inicial, especificar las actividades y acciones que tiene lugar a lo largo del tiempo.
  5. Modelar las acciones complicadas o los conjuntos de acciones que aparezcan muchas veces como llamadas a diagramas de actividades aparte (submáquinas).
  6. Representar los flujos que conectan las acciones y los nodos de actividad
     - Comenzar con los flujos secuenciales del FT.
     - Después considerar las bifurcaciones y fusiones.
     - Por último, tener en cuenta las divisiones y uniones.
  7. Representar los objetos importantes involucrados en el FT.
     - Si es necesario, mostrar sus valores y estado cuando cambien.

Apéndice A: Modelado – Flujo de Trabajo

- **Ejemplo** de Flujo de Trabajo.
  Negocio de Venta.

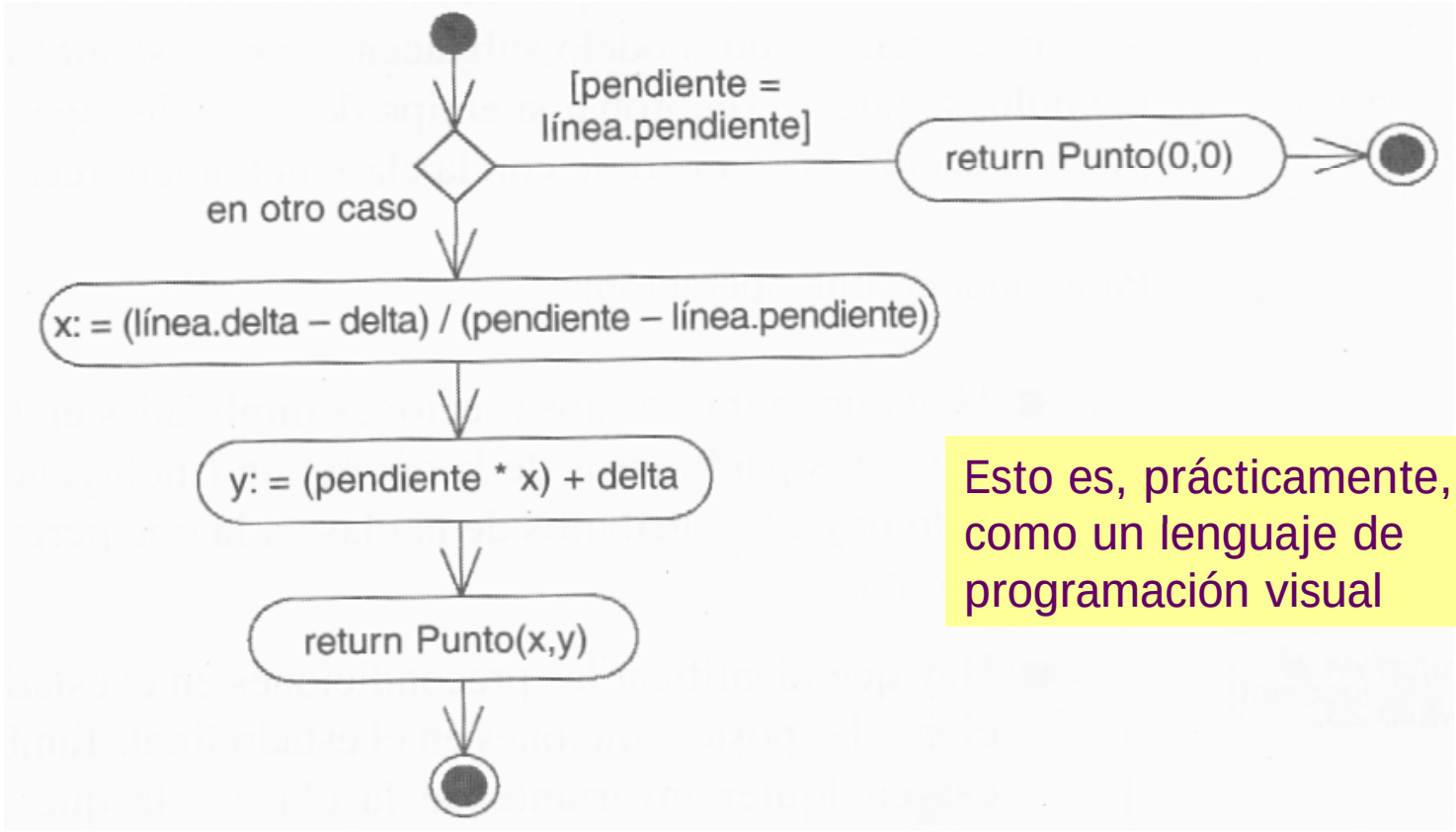


Apéndice A: Modelado – Operación

- Los diagramas de actividades se pueden emplear para representar el **diagrama de flujo** de las acciones de una **operación**.
- Para modelar una operación:
  1. Reunir las abstracciones importantes de la operación (parámetros, tipos, retornos, atributos de la clase, clases vecinas, etc.)
  2. Identificar las precondiciones en el estado inicial y postcondiciones en el estado final. También los invariantes de la clase, que se deben cumplir mientras se ejecuta la operación.
  3. Comenzando por el estado inicial, especificar las actividades y acciones, representándolas como nodos de actividad o acciones.
  4. Usar bifurcaciones y fusiones cuando se necesiten caminos alternativos e iteraciones.
  5. Usar divisiones y uniones para especificar flujos paralelos (sólo si la operación es de una clase activa).

Apéndice A: Modelado – Operación

- Ejemplo de Operación. Intersección en la Clase Línea.



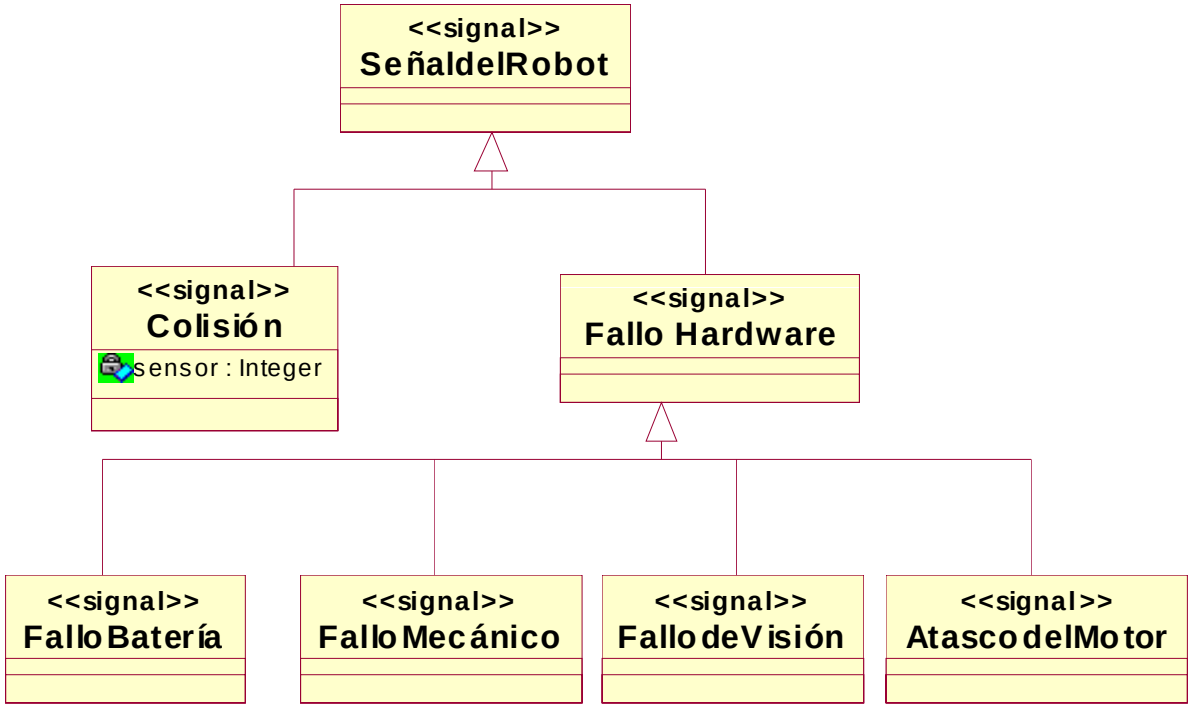
Esto es, prácticamente, usar UML como un lenguaje de programación visual

Apéndice A: Modelado – Familia de Señales

- Para modelar una **familia** (jerarquía) **de señales**:
  1. Considerar todos los tipos diferentes de señales a las cuales puede responder un conjunto de objetos activos.
  2. Buscar los tipos frecuentes de señales y colocarlos en una jerarquía de generalización/especialización por medio de herencia.
  3. Buscar, en las máquinas de estado de los objetos activos, dónde es posible utilizar el polimorfismo.
     - Si hay una transición que se dispara por una “señal padre”, también será disparada con una señal hija.
     - Hay herencia de sus atributos y polimorfismo de sus operaciones.

Apéndice A: Modelado – Familia de Señales

- Ejemplo de una Familia de Señales. Señales de un Robot Autónomo.

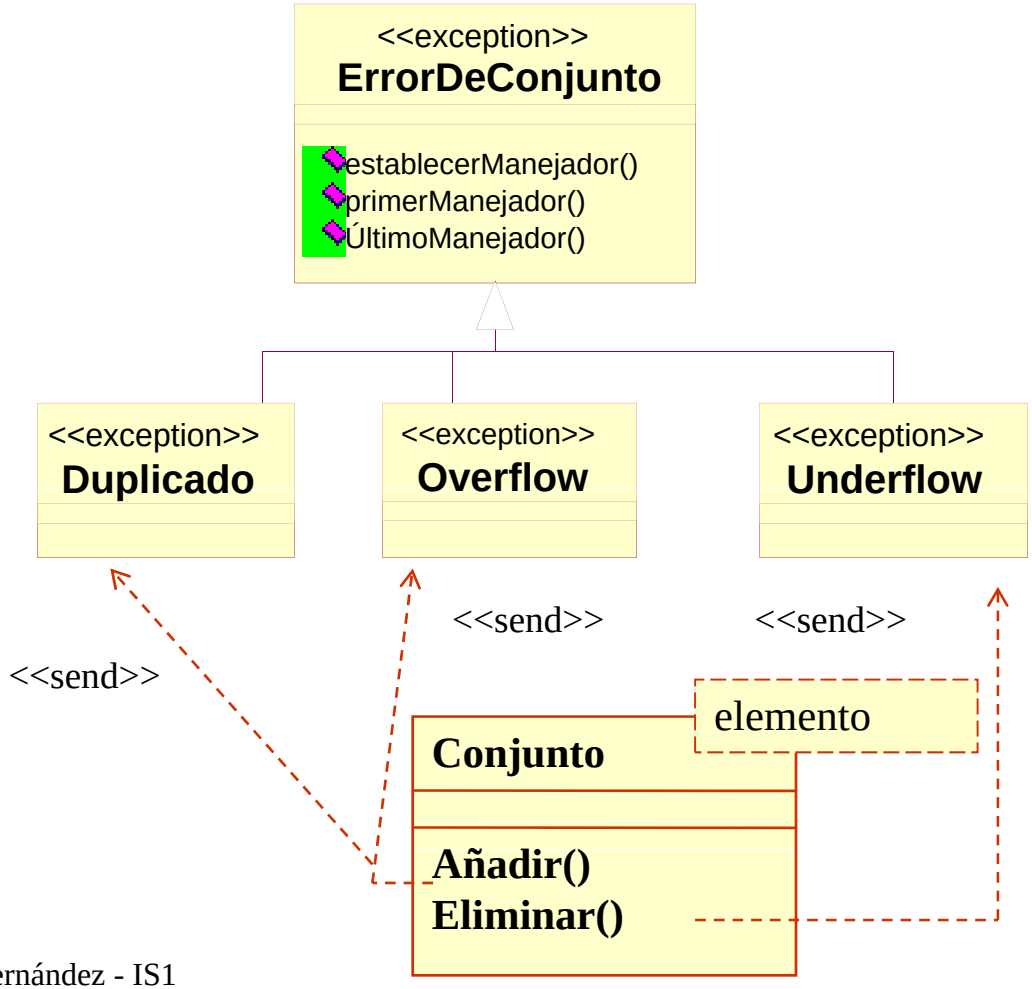


Apéndice A: Modelado – Excepciones

- Al modelar el comportamiento de una clase o interfaz es importante modelar las situaciones anormales (**excepciones**) que pueden producir sus operaciones.
- En UML 2 las excepciones son un tipo especial de evento, que puede modelarse como una señal.
- Para modelar excepciones:
  1. Considerar las excepciones que se pueden producir en cada clase e interfaz, y con cada operación de estos elementos.
     - Pensar lo que puede ir mal y modelarlo como señales entre objetos.
  2. Organizar esas señales en una jerarquía.
     - Introducir excepciones intermedias donde sea necesario.
  3. Especificar las señales de excepciones que puede producir cada operación (dependencias estereotipadas con «send»).

Apéndice A: Modelado – Excepciones

- Ejemplo de Excepciones. Condiciones de error de una biblioteca de clases contenedoras.



Apéndice A: Modelado – Objeto Reactivo

- Al **modelar objetos reactivos** se emplea un diagrama de estados para, básicamente, especificar:
  - Los estados estables en los que puede encontrarse el objeto,
  - Los eventos que disparan una transición entre estados, y
  - Las acciones que tienen lugar durante cada cambio de estado.
- Pasos recomendados:
  1. Elegir el contexto (clase, caso de uso, o sistema global).
  2. Establecer los estados inicial y final, y sus pre y postcondiciones, respectivamente.
  3. Elegir los estados estables del objeto (condiciones que puede satisfacer durante un periodo de tiempo).
     - Empezar por los estados de alto nivel.
     - Después considerar posibles subestados.
  4. Elegir un orden parcial significativo de los estados estables.
  5. Elegir los eventos que pueden disparar una transición y modelarlos como disparadores de las transiciones.

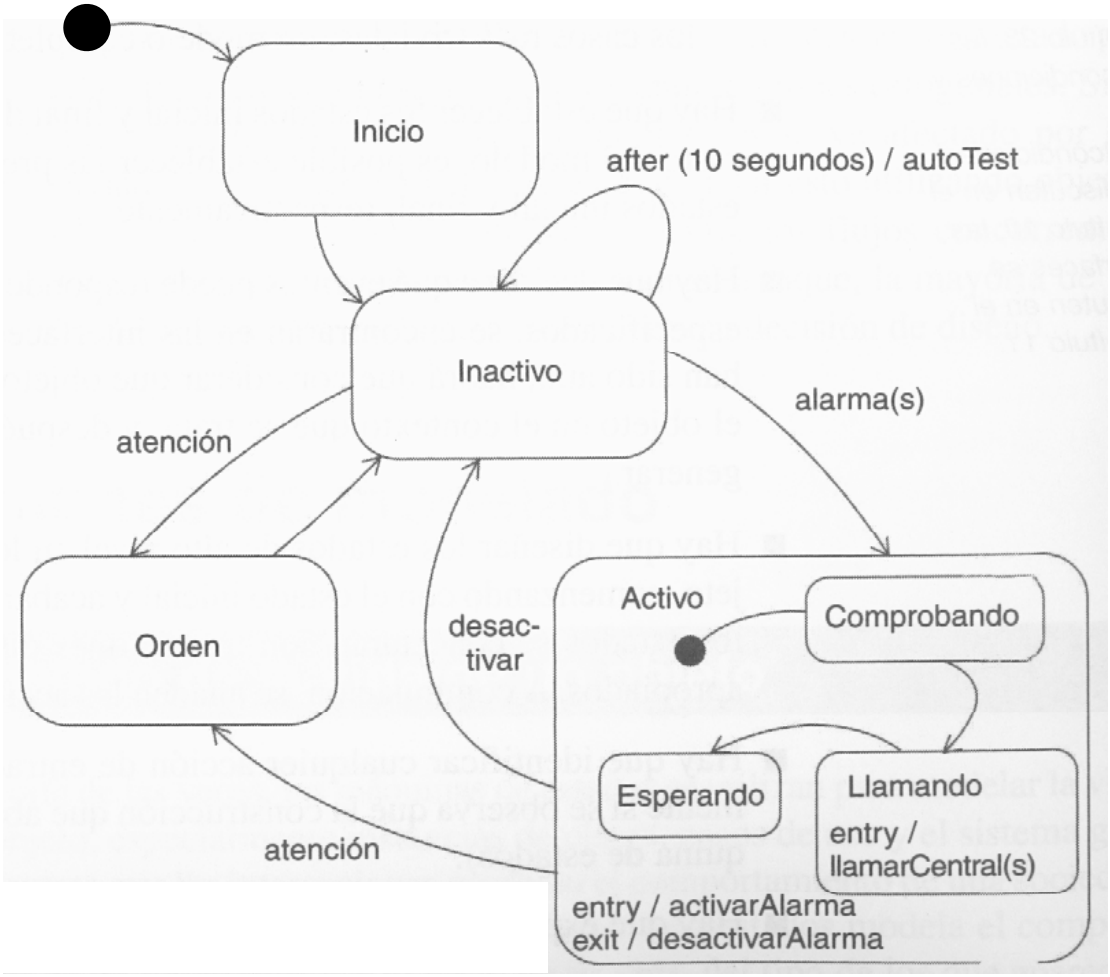
Apéndice A: Modelado – Objeto Reactivo

- Pasos recomendados para **modelar objetos reactivos**: (cont.)

6. Asociar acciones a dichas transiciones y/o a los estados.
7. Considerar diferentes formas de simplificar la máquina de estados.
   - Subestados, Bifurcaciones, Divisiones y Uniones, Estados con historia.
8. Comprobar que todos los estados son alcanzables mediante alguna combinación de eventos.
9. Comprobar que ningún estado es un punto muerto del cual no se puede salir.
10. Hacer trazas a través de la máquina de estados para verificar las secuencias esperadas de eventos y sus respuestas.

Apéndice A: Modelado – Objeto Reactivo

- Ejemplo de Vida de un Objeto. Controlador de un sistema de seguridad.



Apéndice A: Modelado – Objeto Reactivo

- Ejemplo de Objeto Reactivo. Analizador de lenguaje de contexto libre para mensajes XML.