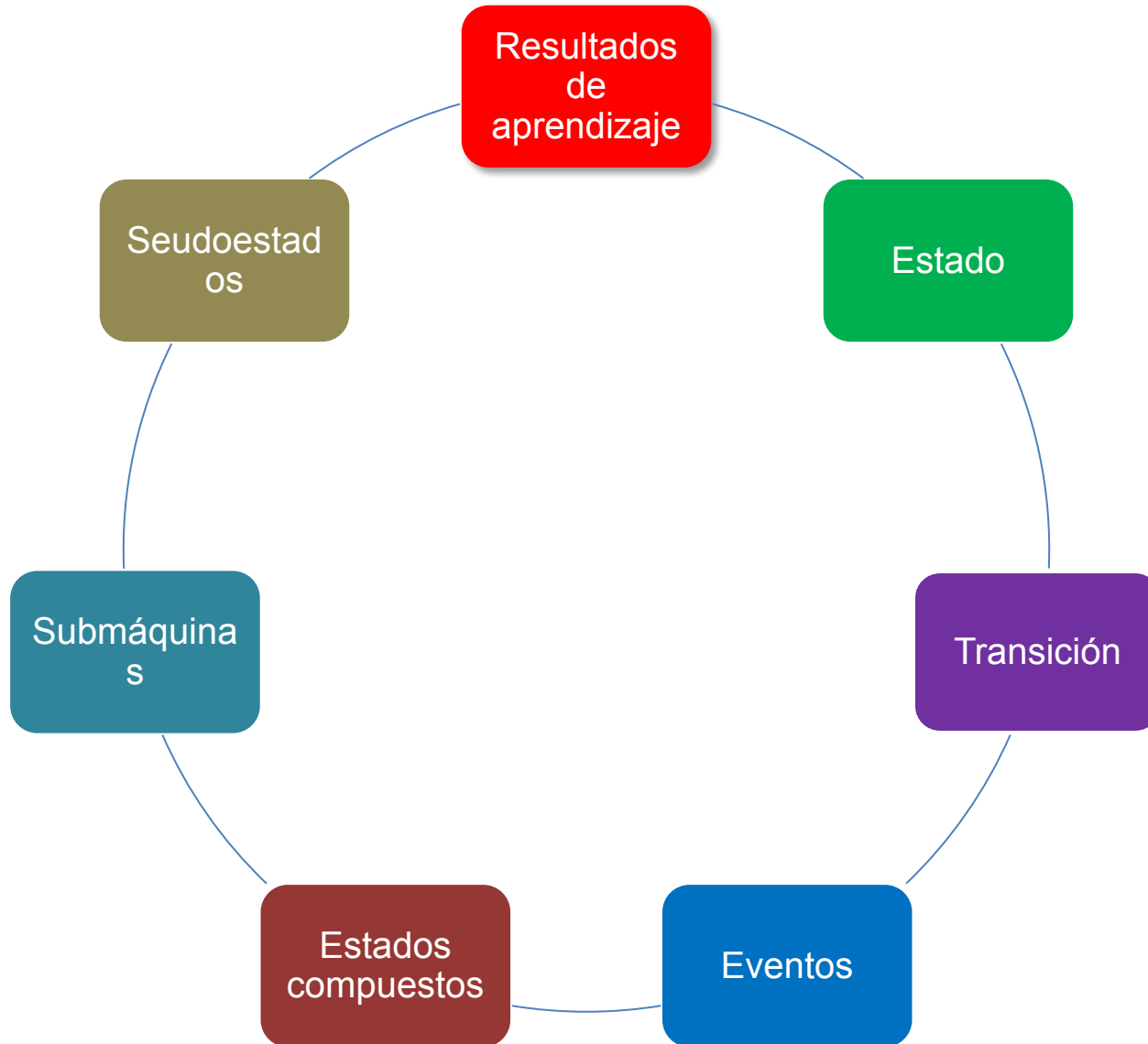


Análisis de sistemas

CIF 5555

2022-1



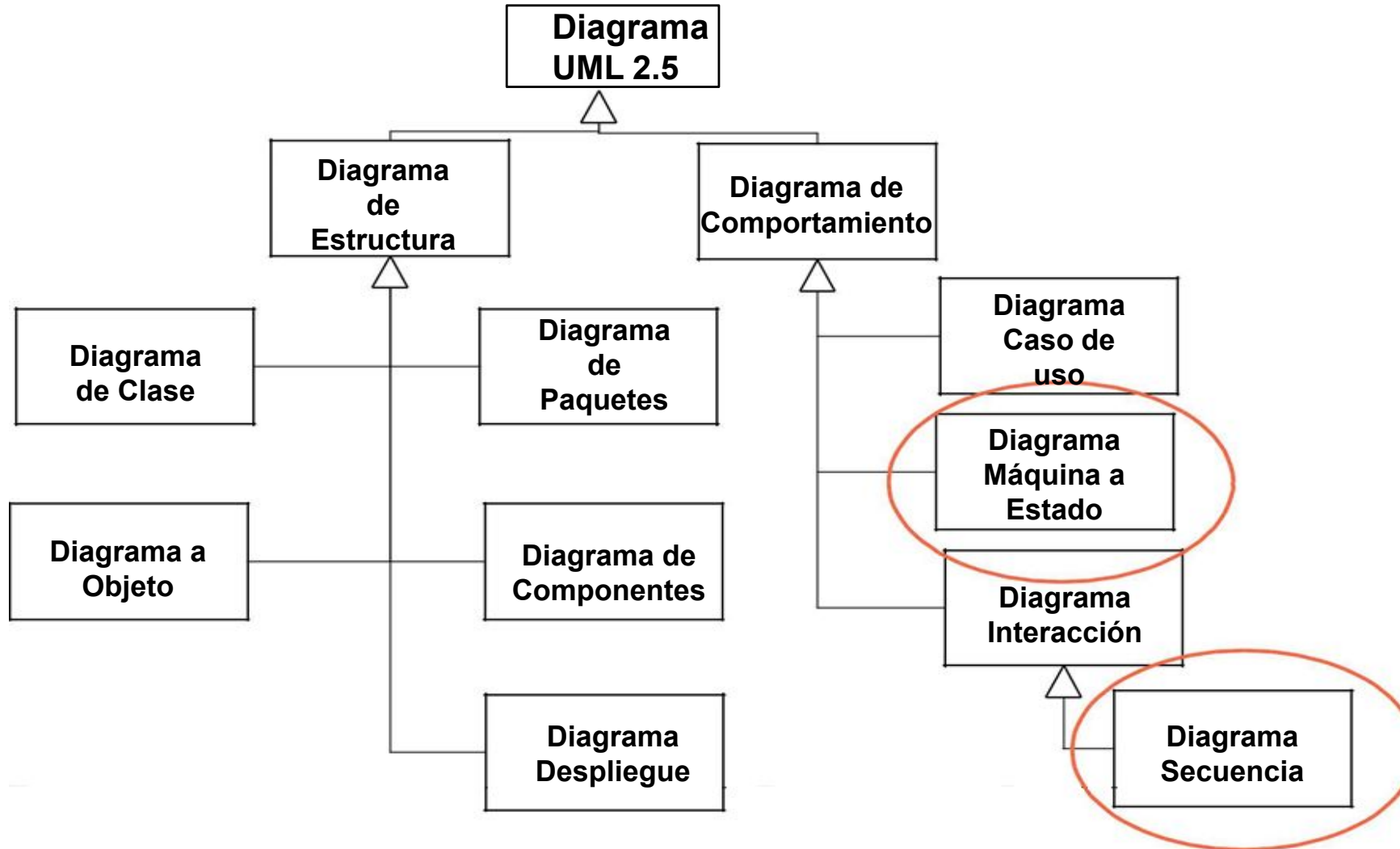
Resultados de aprendizaje

- El alumno debería:
 - Aplicar el proceso de Ingeniería de Requisitos para la elicitación de las necesidades de los clientes y usuarios

Diagramas de Comportamiento

DIAGRAMA DE MÁQUINA A ESTADO

Diagramas conocidos de UML



Máquinas a estado

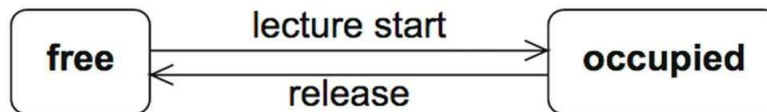
- Una máquina de estado describe el comportamiento dinámico de instancias de un clasificador (por ejemplo, objetos de instancia de una clase).
- Para construir una máquina de estados, necesitamos identificar los estados significativos en los que se puede encontrar un objeto durante su vida.
- Además, debemos describir cómo desde cada uno de estos estados el objeto puede pasar (transición) a otro.
- Las transiciones ocurren en respuesta a la ocurrencia de un evento. Los eventos son típicamente;
 - mensajes enviados por otros objetos
 - eventos generados internamente
- Una máquina de estados se representa con un grafo de estados y transiciones, asociado a un clasificador.

Estado

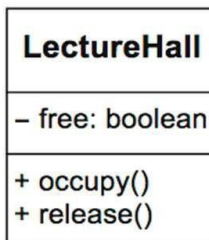
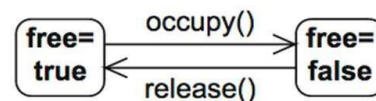
- Un estado es un conjunto de valores (de algunas variables significativas) de un objeto:
 - Es una abstracción del estado concreto del objeto (caracterizado de los valores de todas las variables)
 - Representa un estado significativo
 - Se caracteriza por dar la misma respuesta cualitativa a los eventos. eso puede pasar.
- Un estado tiene un nombre único
- Un estado puede ser compuesto (más tarde)

Es importante el nivel de detalle

- Para modelar un aula



- que luego se especificará en el nivel de implementación



```
class LectureHall {
    private boolean free;
    public void occupy() {
        free=false;
    }
    public void release() {
        free=true;
    }
}
```


Sintaxis básica

Los estados se representan con rectángulos redondeados

- El disco negro marca el comienzo. No es un estado real, sino un marcador que apunta al estado desde el que empezar.
- El disco negro con borde (nodo final) indica la terminación.
- Pueden aparecer en cualquier número dentro de un diagrama.

Estado :



Estado inicial :

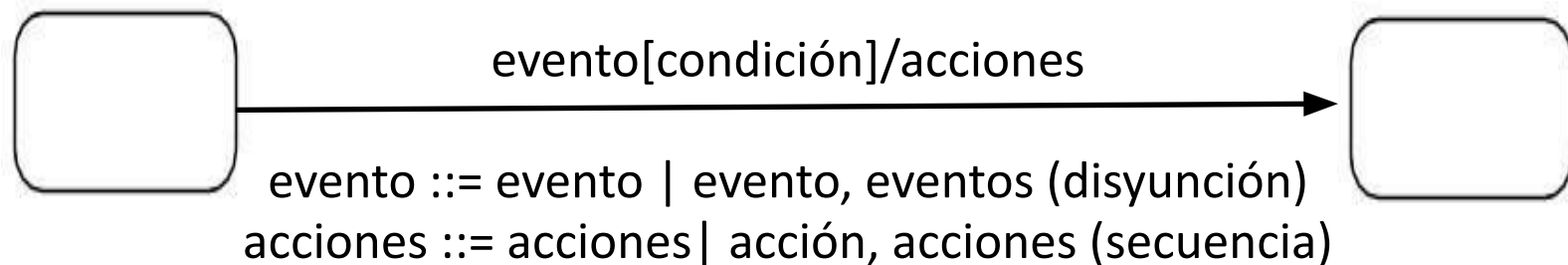


Estado final :



Transición

- Una transición conecta dos estados juntos, se representa con una flecha
- La salida de un estado define la respuesta del objeto a la ocurrencia de un evento, se toma solo si la condición es verdadera e involucra la ejecución de las acciones especificadas.

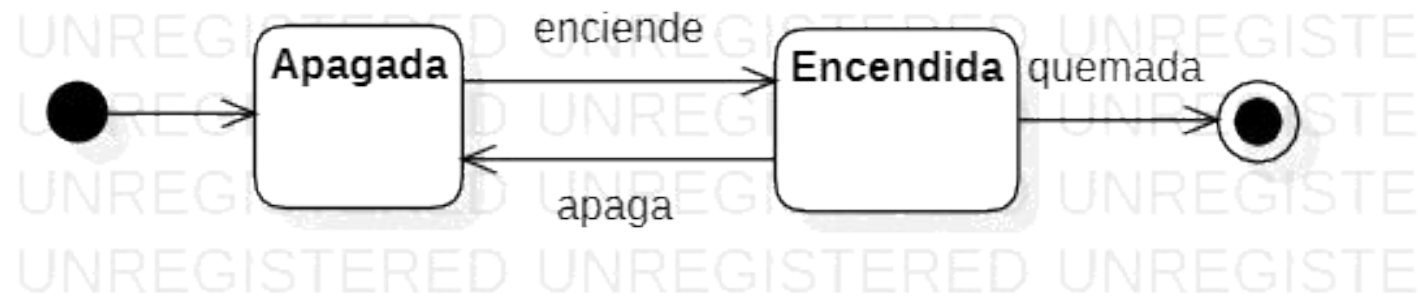


Todos opcionales aunque está bien que esté el evento.

Solo en las transiciones de finalización (posteriores) el evento no es necesario.

Ejemplo estado de una lámpara

Describimos la vida de una lámpara



Lampara
encendida: boolean = false
enciende() apaga()

NOTA: los eventos (aunque no todos) corresponden a las operaciones de la clase

Evento

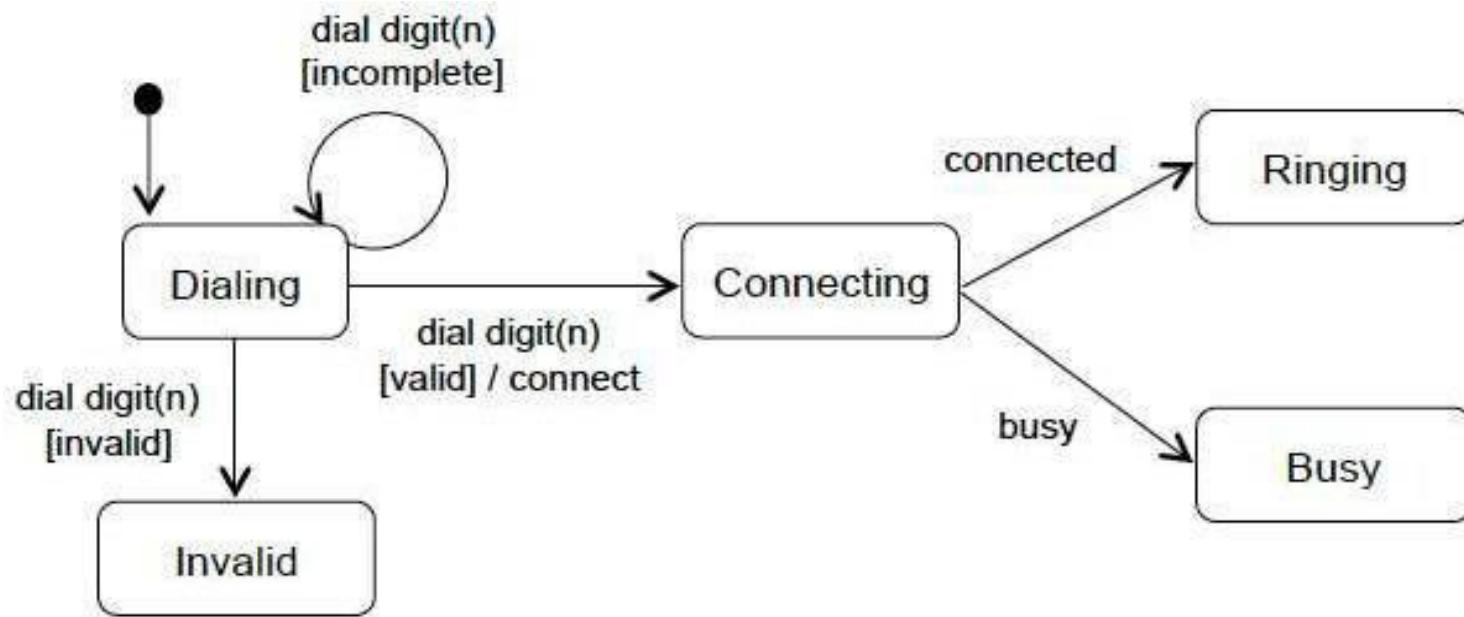
- Un evento es la ocurrencia de un fenómeno ubicado en el tiempo y el espacio.
- Un evento ocurre instantáneamente
- Modele algo como un evento si tiene consecuencias
- Los eventos que llegan a un estado para el que no hay una transición etiquetada con ese evento se ignoran
- Se permite el no determinismo: un evento puede desencadenar múltiples transiciones:
 - Si las dos transiciones salen del mismo estado, se elige una de forma no determinista

Tipos de evento

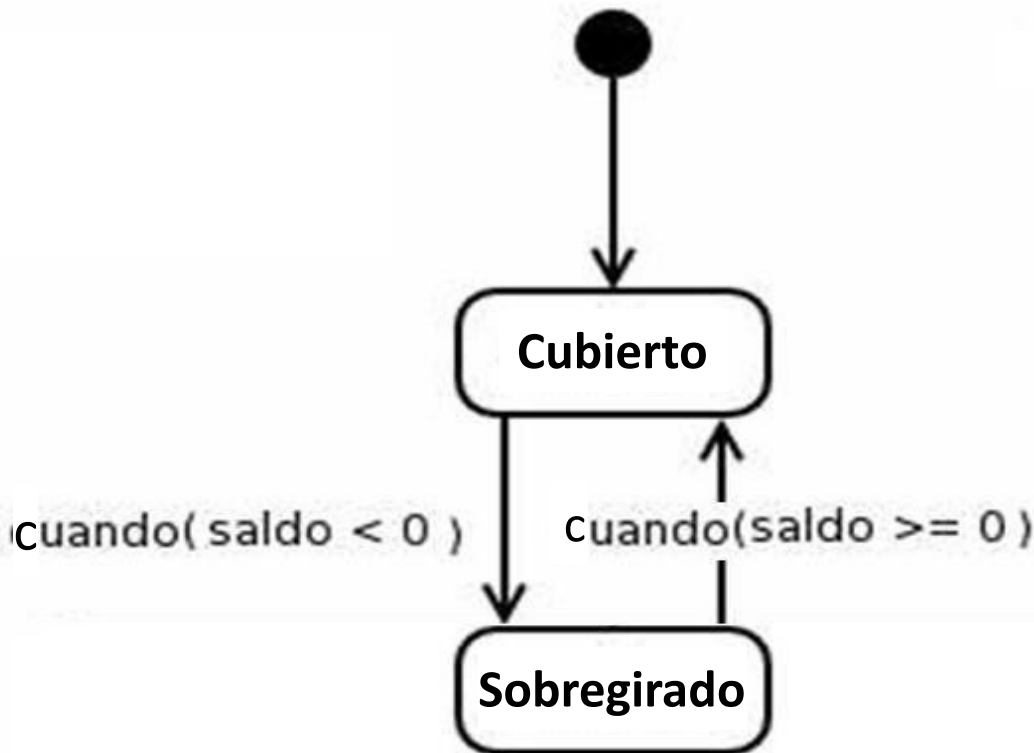
- Operación o señal **op(a:T)**
 - la transición se habilita cuando el objeto (en ese estado) recibe una llamada de método / una señal con los parámetros (a) y tipo (T) (los parámetros son opcionales)
- Evento de variación **cuando(exp)**
 - la transición se habilita tan pronto como la expresión se vuelve verdadera
 - la expresión puede indicar un tiempo absoluto o una condición en las variables
 - a menudo en inglés: when (exp)
- Evento temporal **despues(time)**
 - la transición se habilita después de que el objeto ha estado detenido "time" en ese estado
 - a menudo en inglés: after (time)

Evento operación o señal

- Operaciones de la clase teléfono (marcar dígito (n)) o señales (ocupado y conectado)

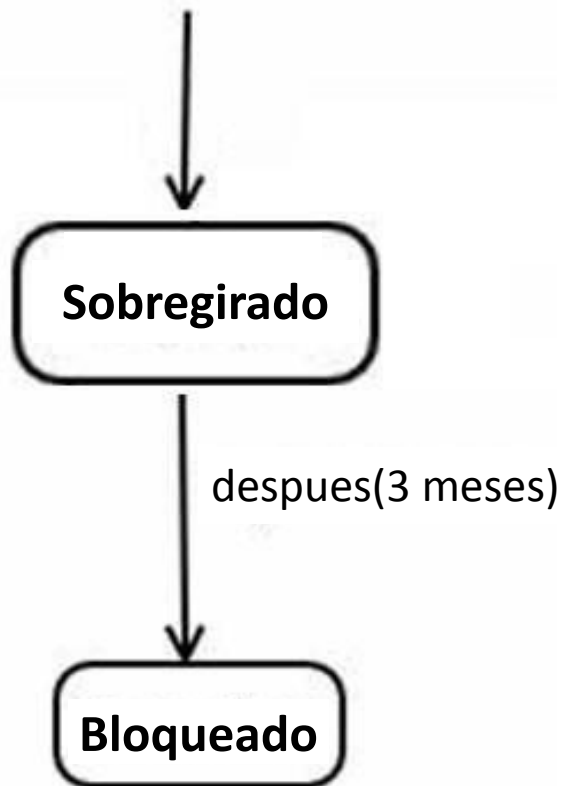


Evento de variación (ejemplo)



- Un evento ocurre de modo instantáneo
- Una condición no es instantánea
- el momento en que se vuelve verdadero es instantáneo

Eventos temporales (ejemplo)

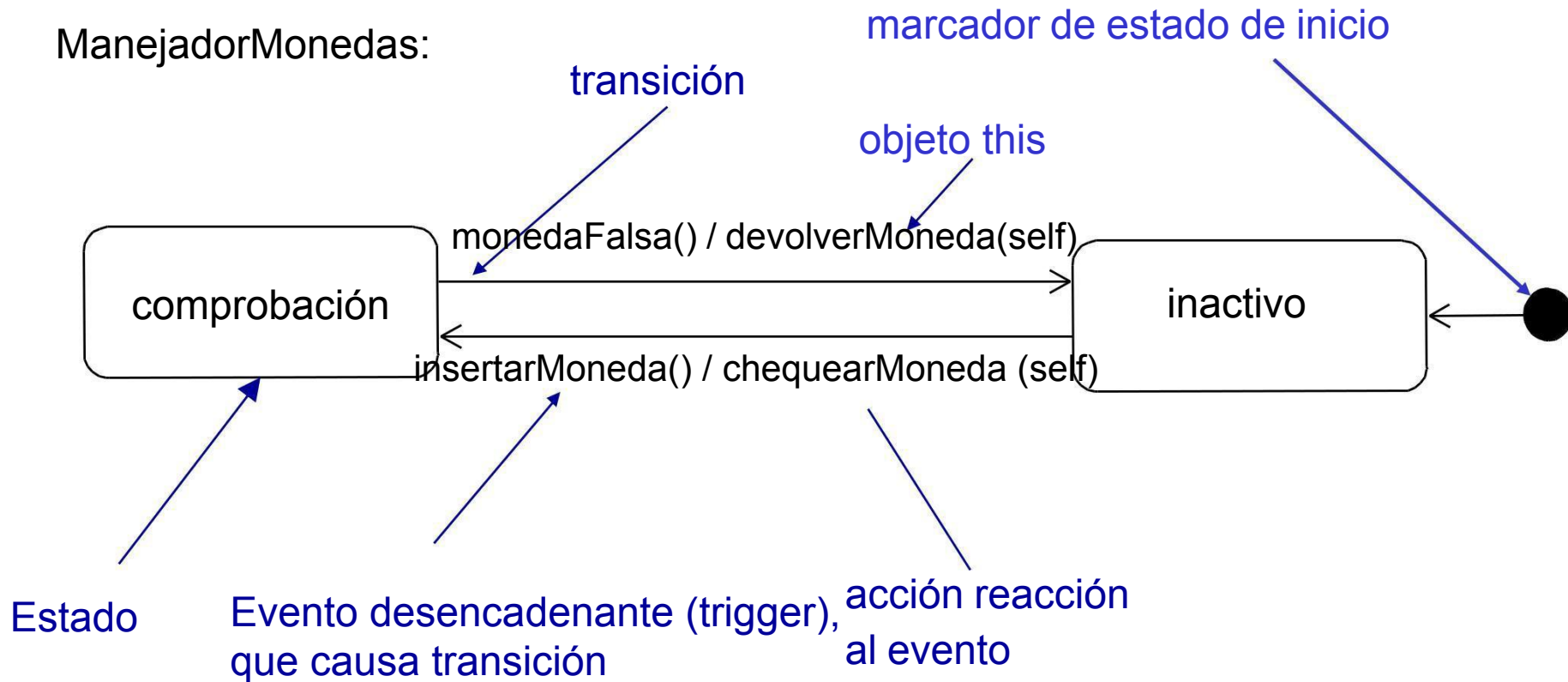


Una vez que el objeto ha estado 3 meses en el estado Sobregirado, pasa al estado Bloqueado

Diagrama de máquina a estado

Para clase

ManejadorMonedas:

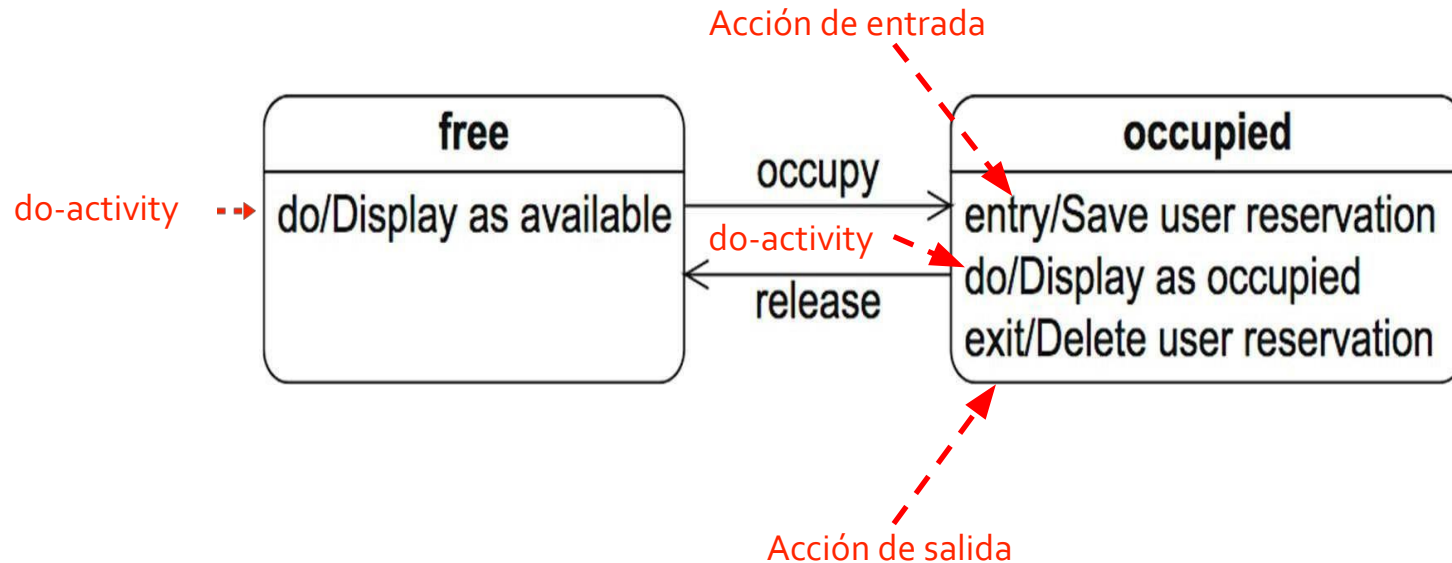


Transiciones y actividades internas

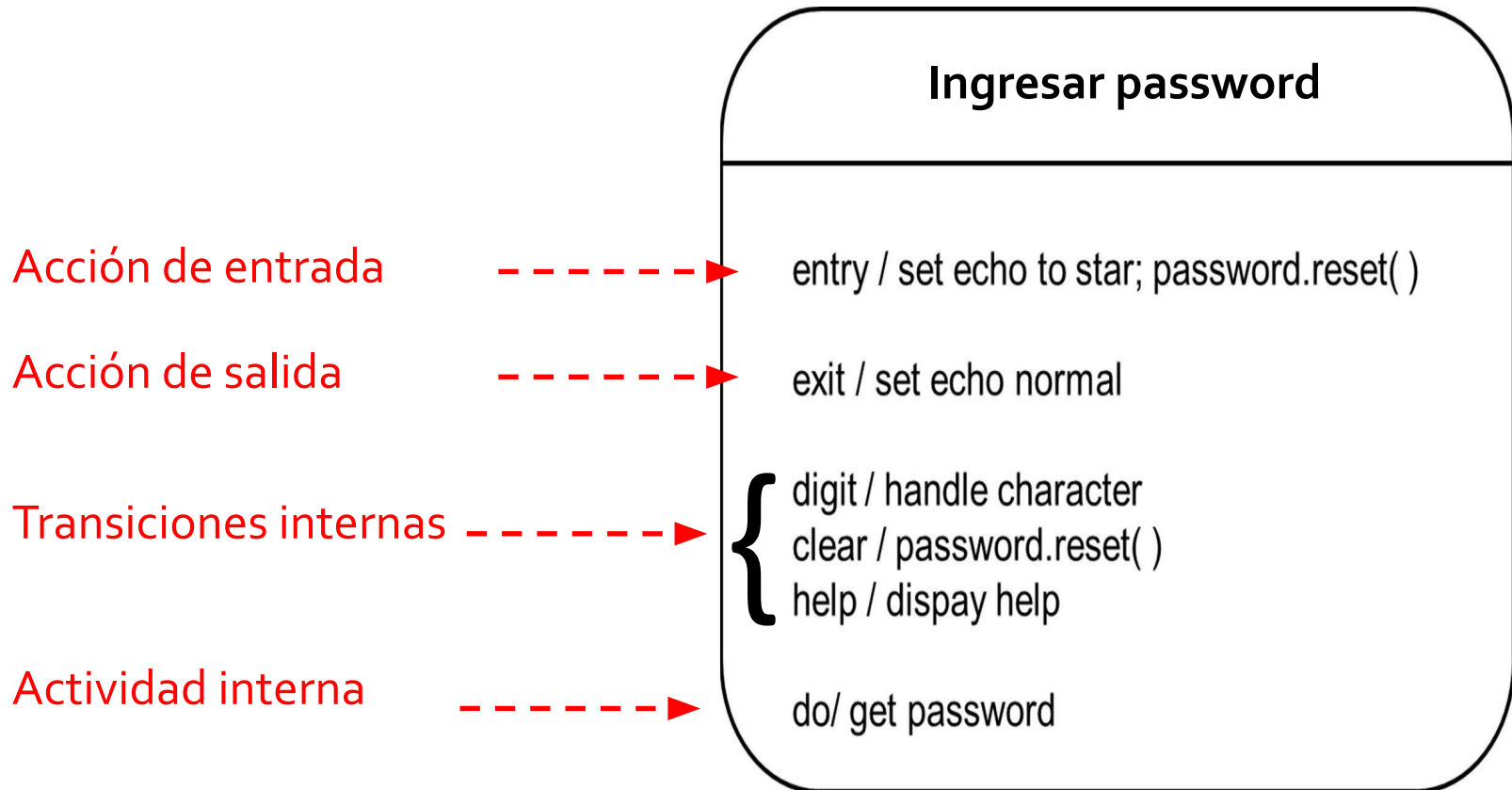
- Transición interna: respuesta a un evento que solo causa la ejecución de acciones. Ejemplos:
 - **Acción de entrada**: ejecutada al entrar en un estado
 - **Acción de salida**: se ejecuta cuando se sale de un estado
 - **Transición interna**: respuesta a un evento
- **Actividad interna (Do-activity)**: se ejecuta de forma continua mientras el objeto está en ese estado (sin la necesidad de un evento desencadenante), a diferencia de todas las demás acciones que son atómicas:
 - consume tiempo
 - se puede abortar (cuando un evento hace salir del estado)

Ejemplo

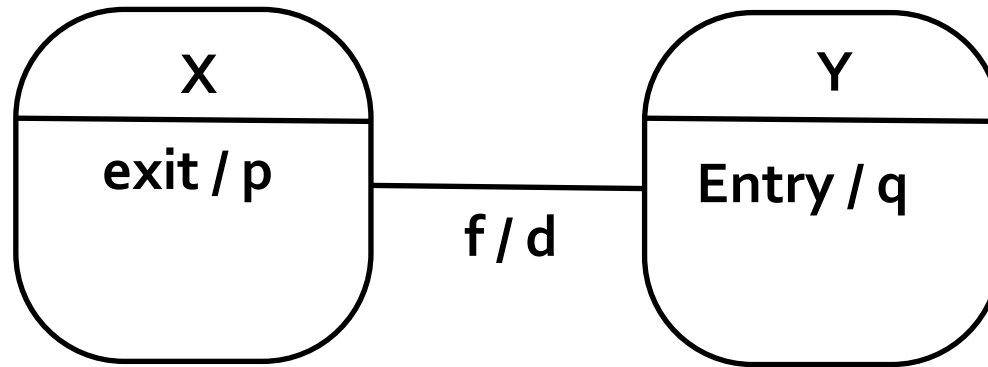
Volvemos al ejemplo del aula:



Sintaxis



Ejemplos



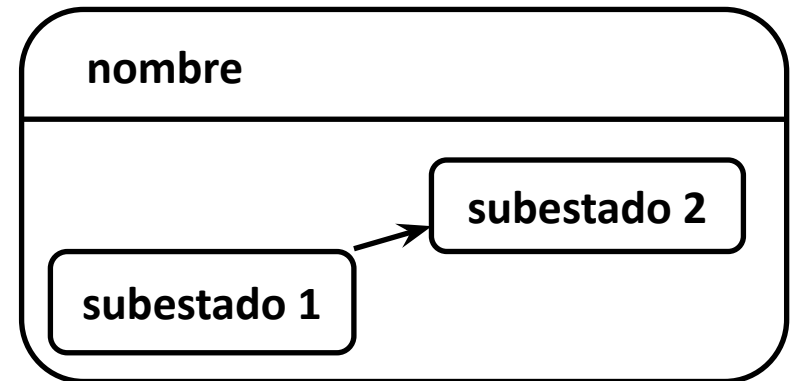
effective result: f / p; d; q

Sobregirado

cuando(saldo < limiteSobregiro) / notificaAdministrador

Estados compuestos

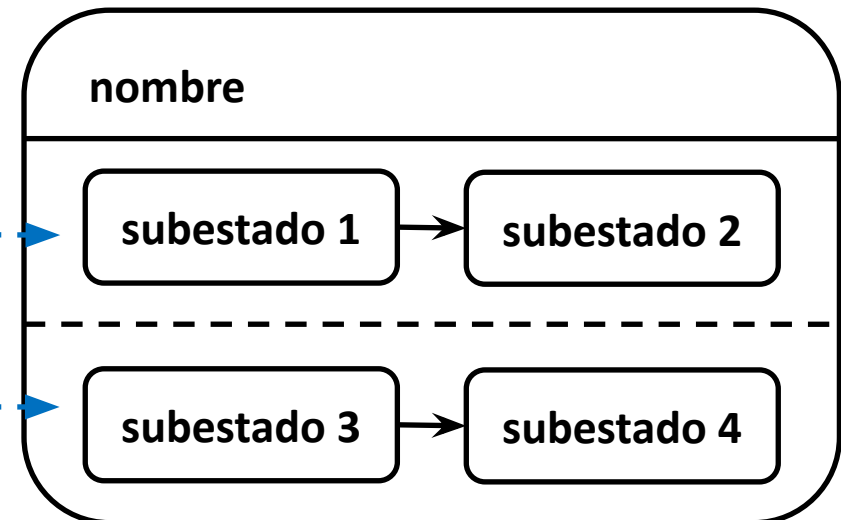
- Compuesto secuencial
 - Un subestado activo en todo momento



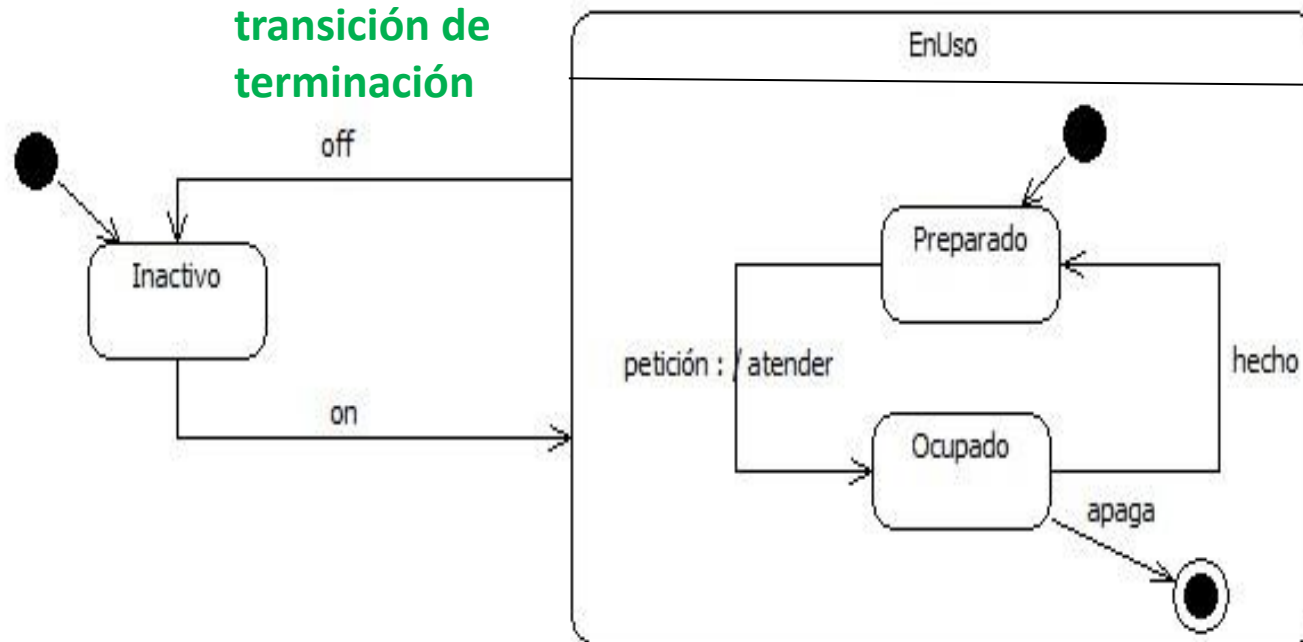
- Compuesto paralelo:
 - subestados activos al mismo tiempo
 - uno por región

región 1 - - - ->

región 2 - - - ->



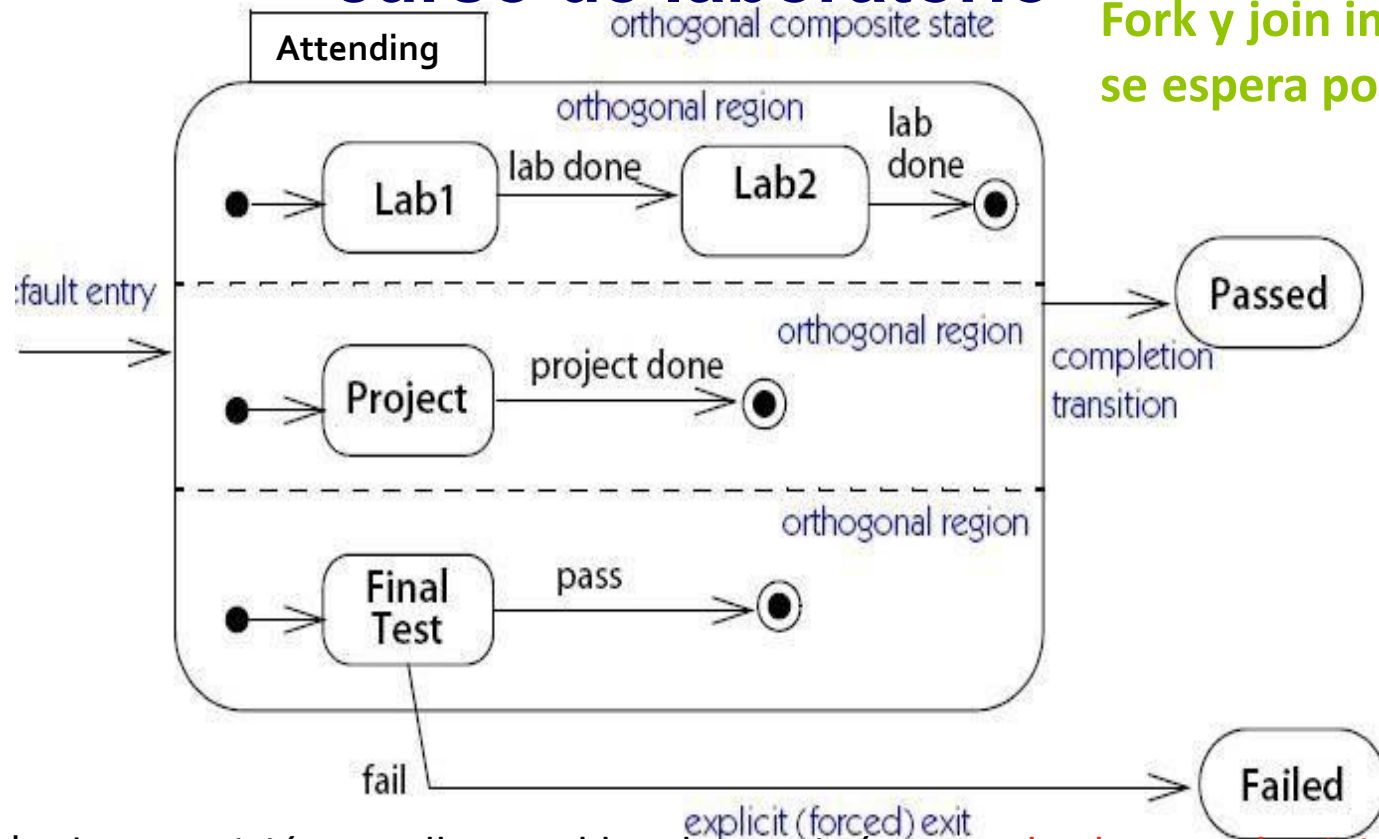
Ejemplo estados compuesto secuencial



- Cada transición que llega al borde continúa en el estado inicial.
- Desde el estado final (después de las salidas) continuamos en **transición de finalización**
- Cualquier transición (no finalizada) que comience desde el borde se considera posible **desde cualquier estado interno**

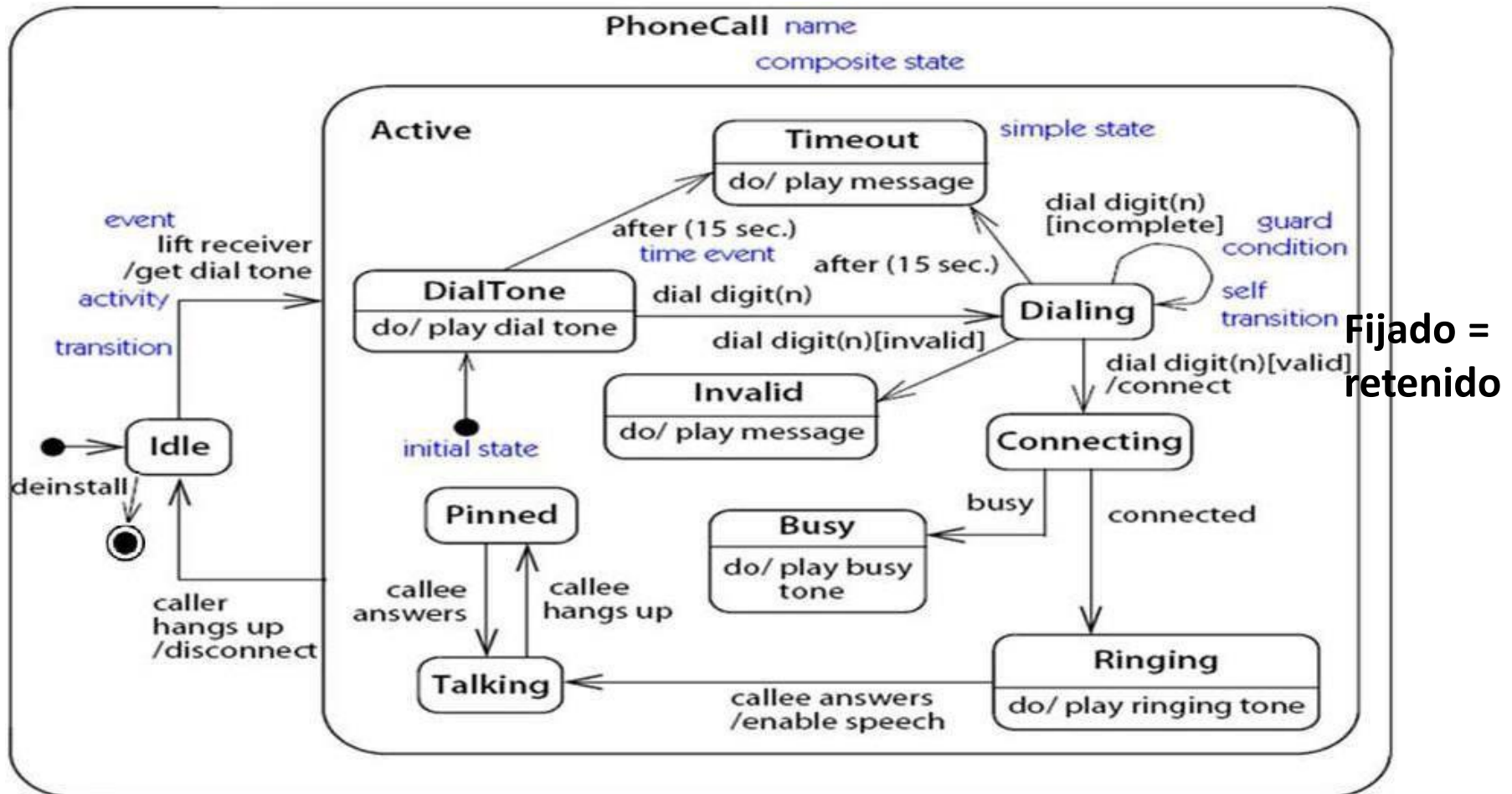
Estado compuesto paralelo (ortogonal): curso de laboratorio

Fork y join implícitas:
se espera por salir.

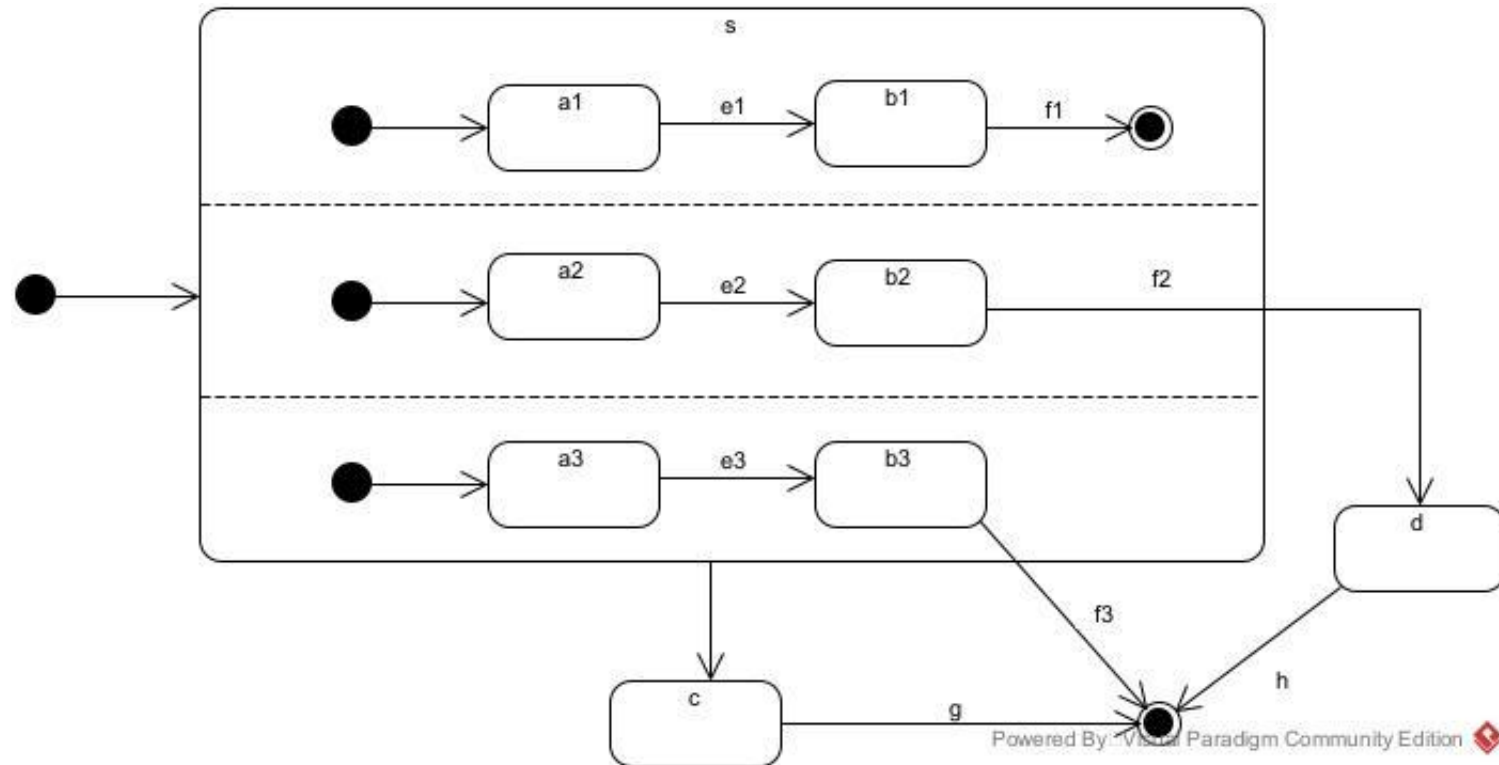


- Cualquier transición que llegue al borde continúa **en todos los estados iniciales**
- Una vez logrado **todos los estados finales** la transición de finalización continúa
- Puede haber transiciones que perforan el borde y están pensadas como posibles **desde el estado interno** al que están vinculadas

Ejemplo: estado compuesto secuencial sin estado final



Ejemplo de estado compuesto paralelo



f1: se termina la región y se queda en s en espera

f2: se sale de s y se pasa a d

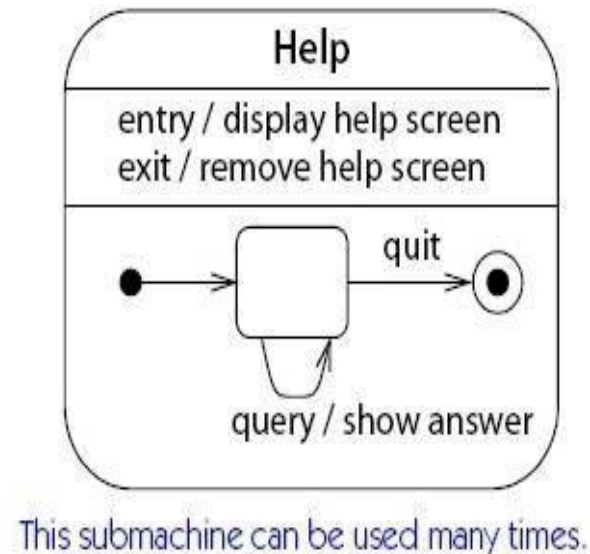
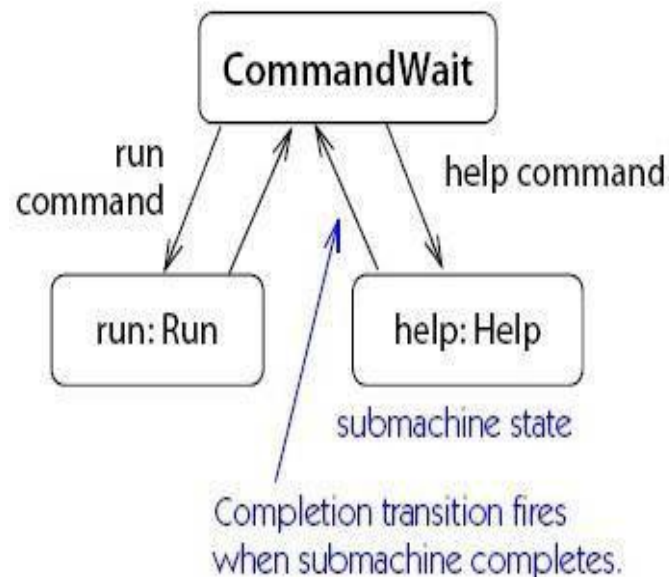
f3: se sale de s y se termina

Al salir de s también se sale de todos los subestados

(La transición de finalización que lleva a c nunca se toma)

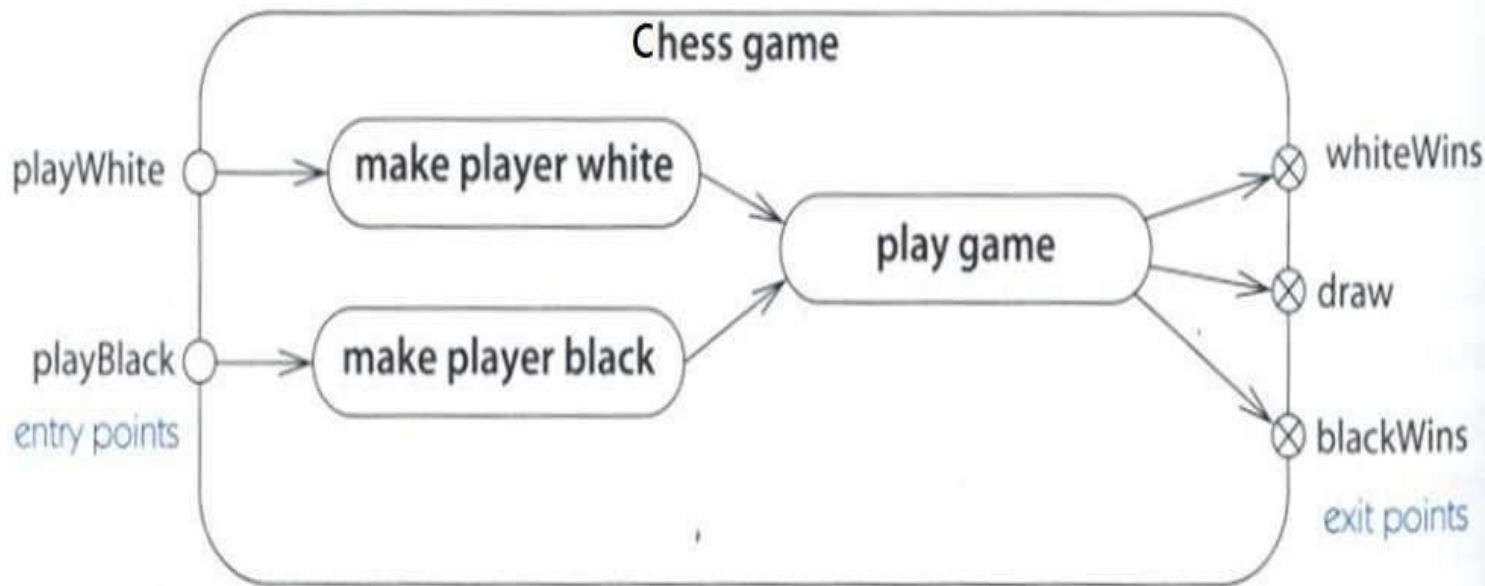
Submáquinas

- Se utiliza cuando se desea describir un estado compuesto en un diagrama separado, por legibilidad o para definirlo de una vez por todas y reutilizarlo en múltiples contextos.
- La submáquina tiene un nombre (tipo), las instancias de uso se indican con nombreInstancia: Tipo

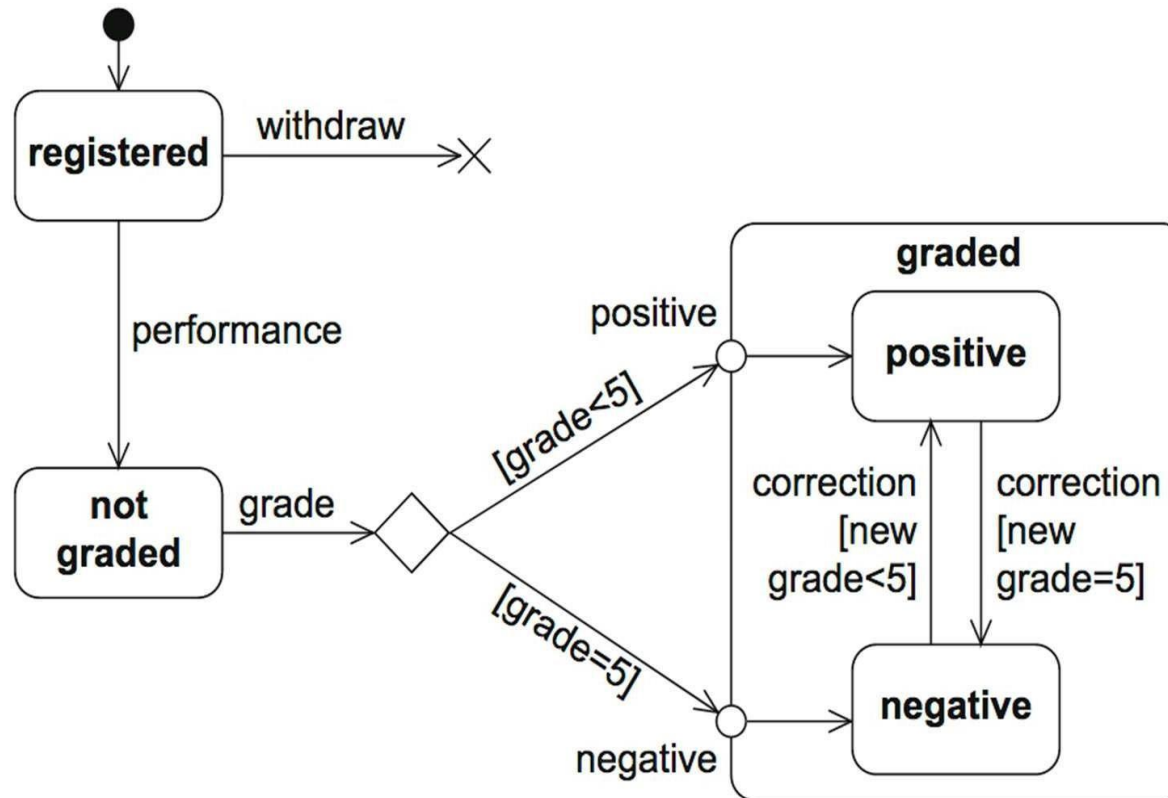


Submáquinas: entry y exit points

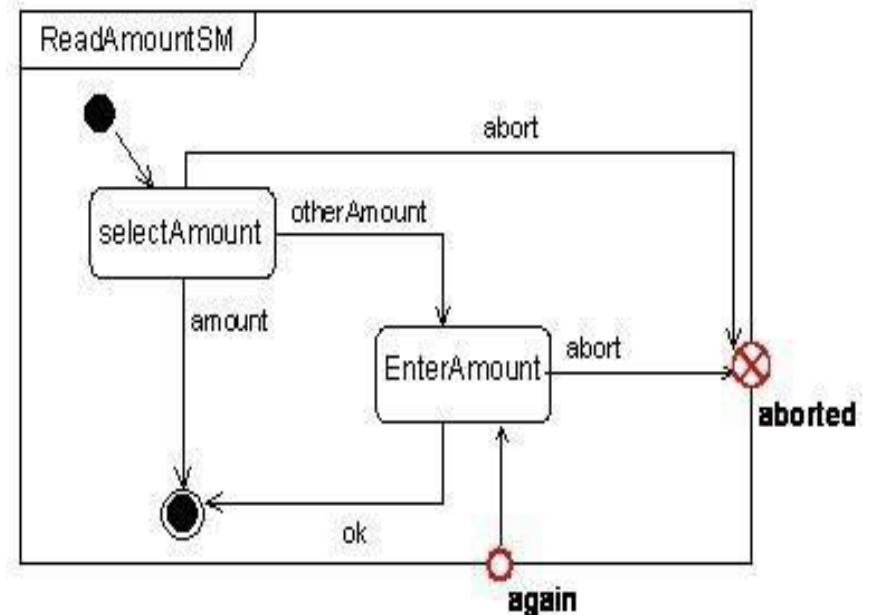
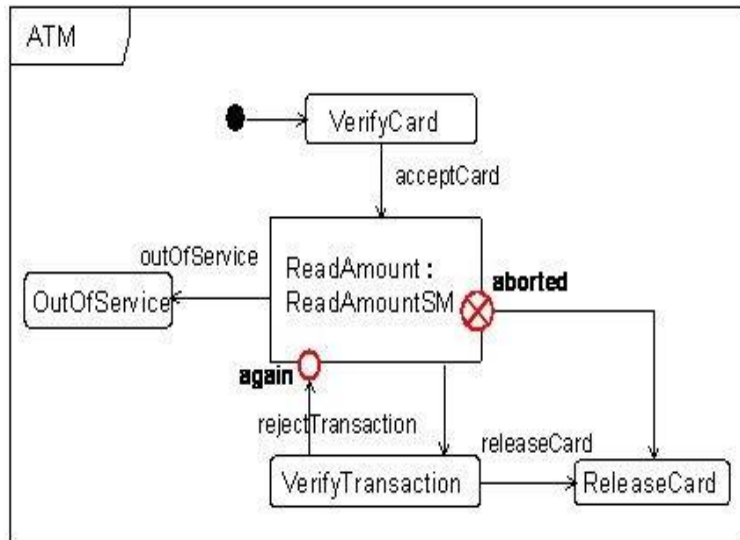
Una submáquina puede definir puntos de entrada y salida que se utilizan para vincular las transiciones de la máquina principal.



Ejemplo



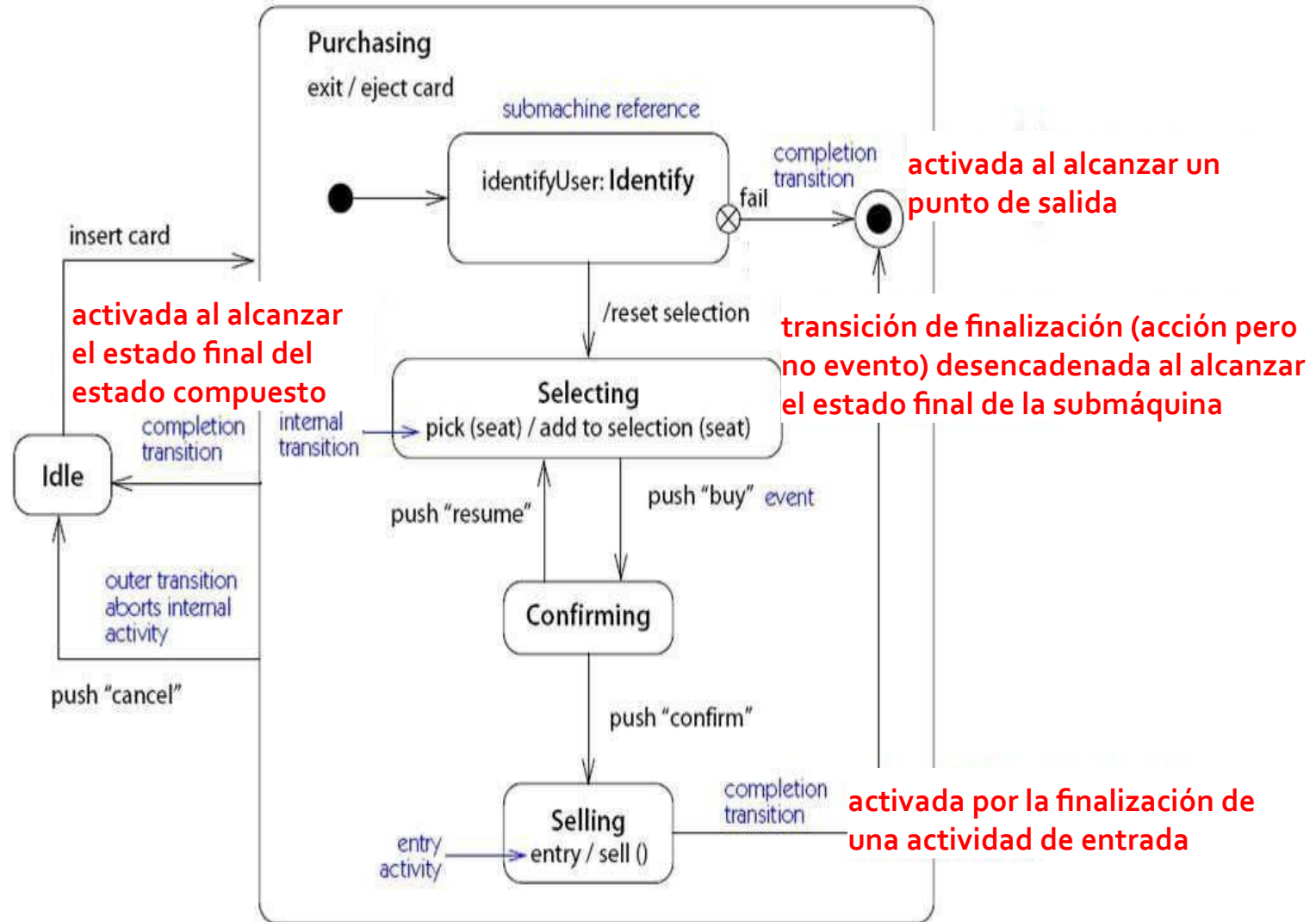
Otro ejemplo



Transiciones de finalización: resumen

- Sin evento, se activan con el cumplimiento
 - De la terminación de una actividad compuesta, es decir, tras el cumplimiento
 - Del estado final en un estado compuesto no ortogonal
 - Estados finales de todas las regiones de un estado compuesto ortogonal
 - De un exit point
 - Al finalizar la actividad de entry y / o do activity (la exit activity se ejecuta cuando se activa la transición de finalización)
 - De un pseudo-estado de unión (lo veremos en un momento)
- Tienen prioridad sobre los eventos normales

Transiciones de finalización: ejemplo

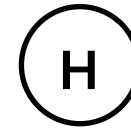


Otros tipos de estado (pseudoestados)

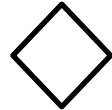
Unión



Historia



Decisión



Inicial



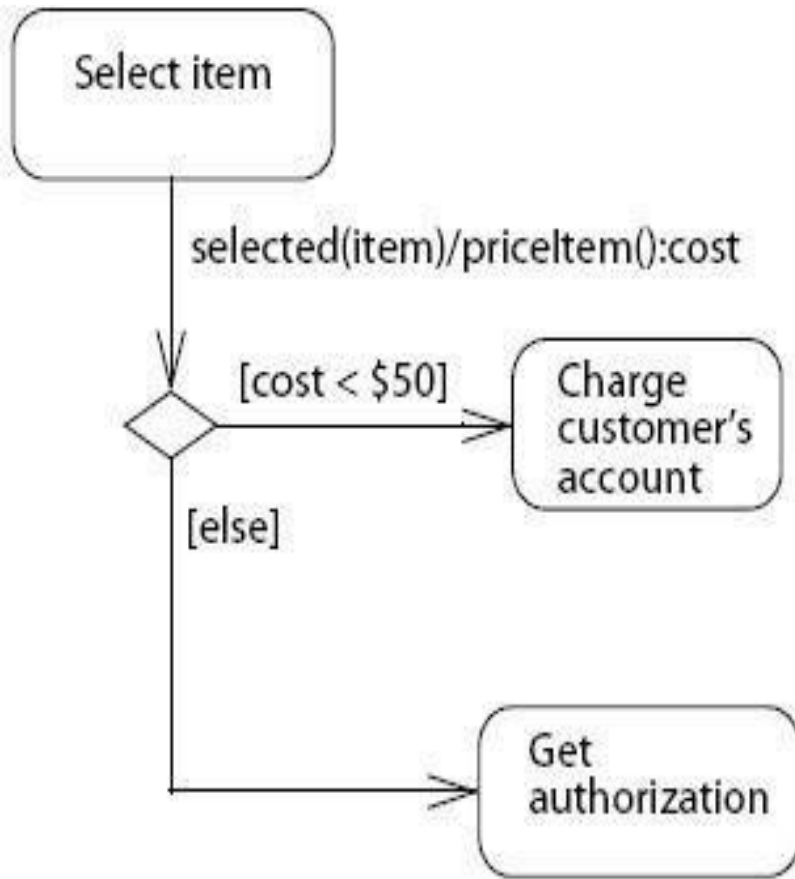
Fork, join



Final

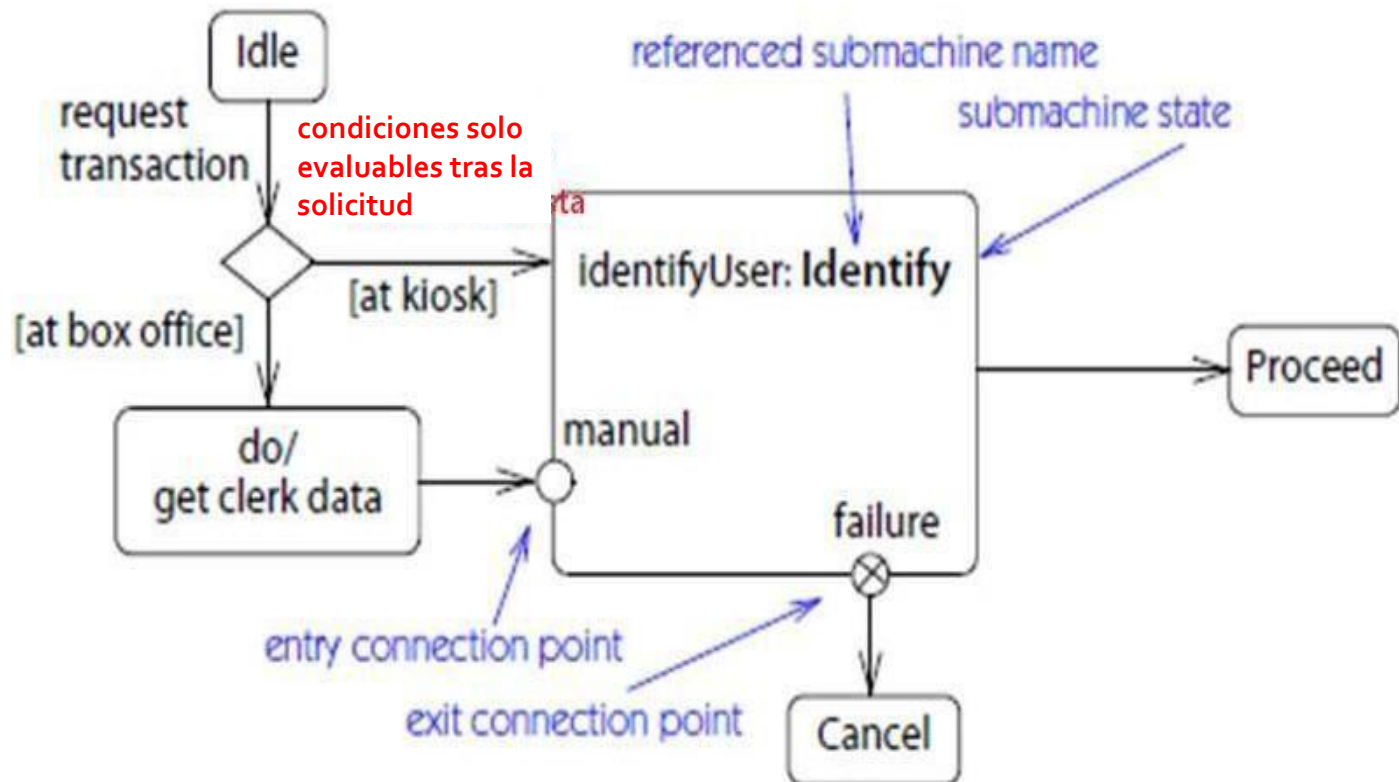


Ejemplo de choice

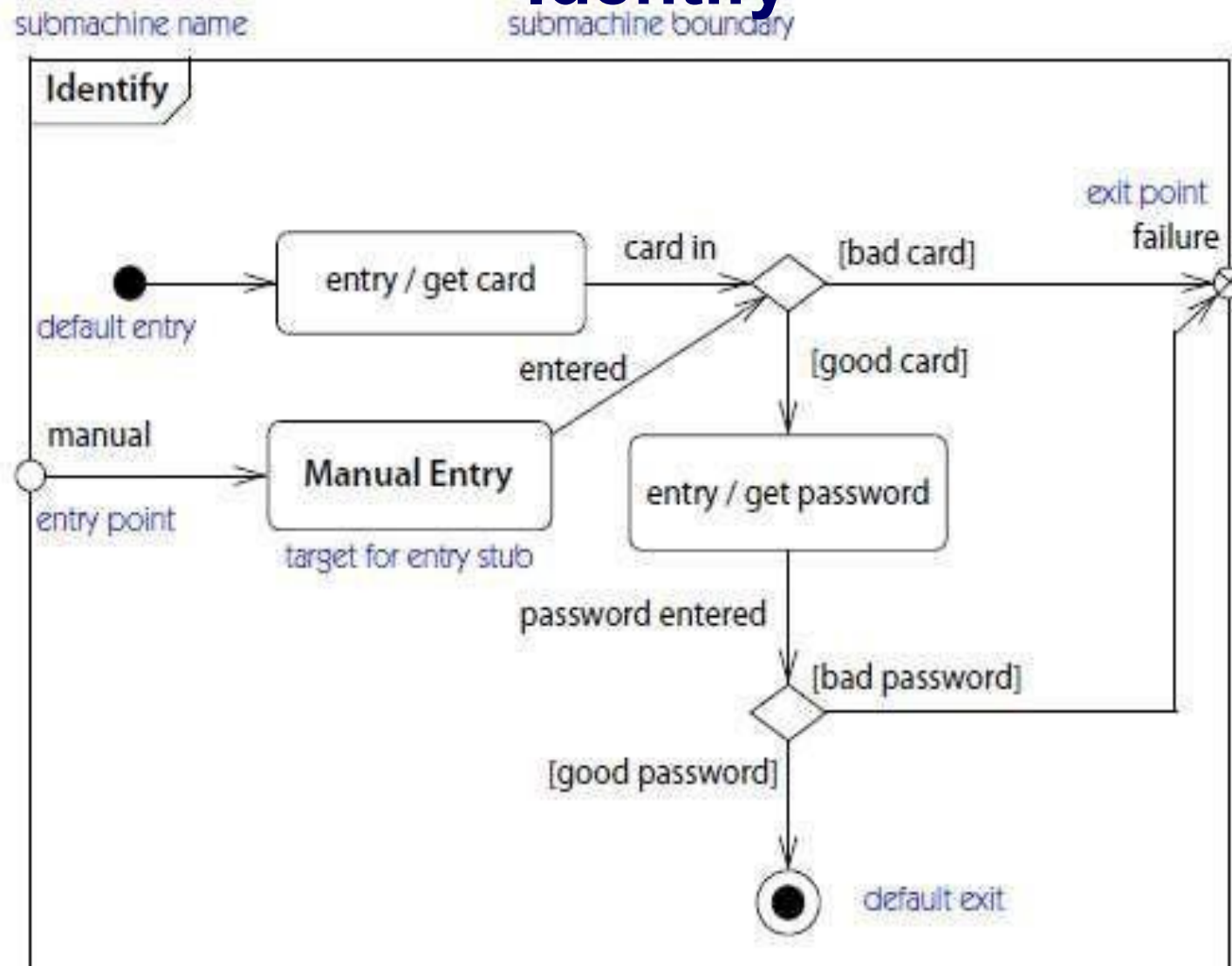


- Condiciones evaluadas dinámicamente
- En cuanto al choice de los diagramas de actividad:
 - La disyunción de los guardias debe ser "true"
 - Se permite el no determinismo

Compra pasajes a cliente con cuenta: identificación por defecto leyendo una tarjeta, manual, con ingreso nombre

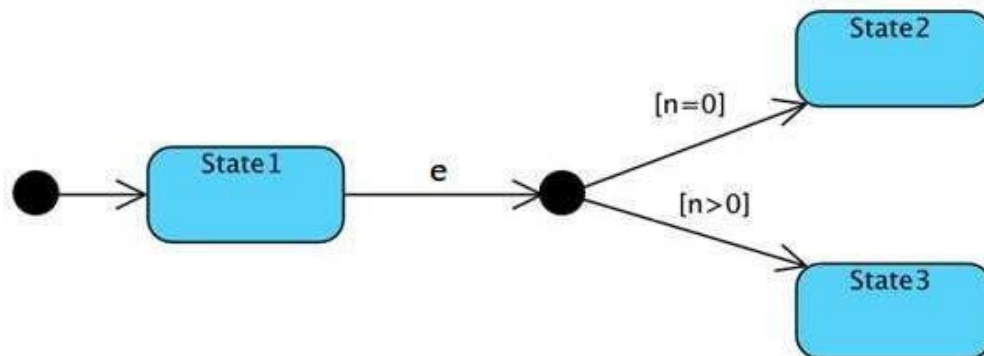


Ejemplo compra pasajes: Submáquina Identify

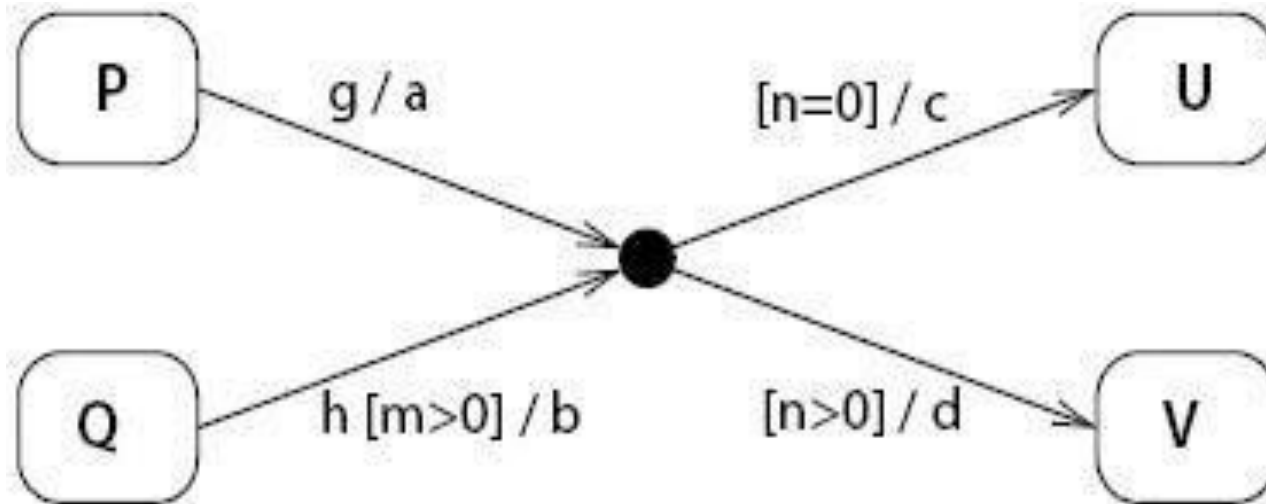


Seudo-estado de unión

- Un pseudo-estado del cual salen y / o entran más transiciones. Cualquier condición se puede evaluar de modo estático (antes del evento e)
- Si $n < 0$, el evento e se ignora y permanece en el estado 1



Ejemplo de Unión



Equivalent transitions:

$P \xrightarrow{g \ [n=0] / a, c} U$

$P \xrightarrow{g \ [n>0] / a, d} V$

$Q \xrightarrow{h \ [m>0 \text{ and } n=0] / b, c} U$

$Q \xrightarrow{h \ [m>0 \text{ and } n>0] / b, d} V$

Ejemplo de uso del pseudo-estado unión

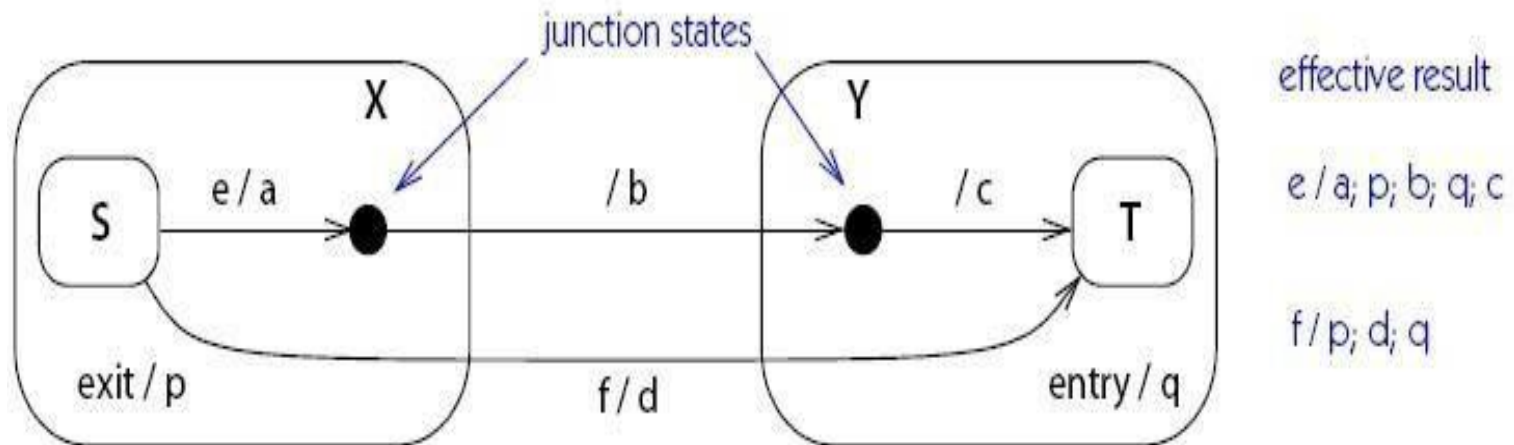
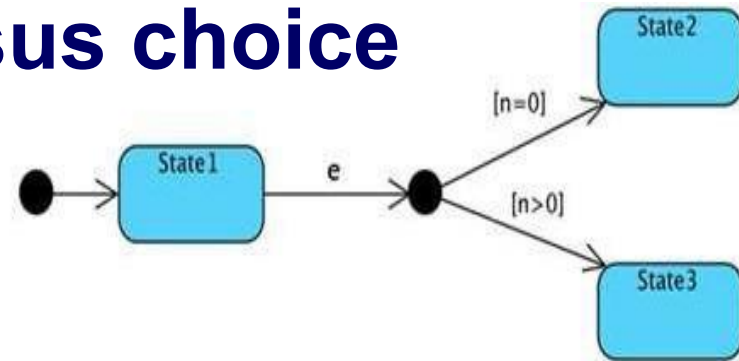


Figure 14-170. *Junction pseudostates with multiple actions*

Unión versus choice

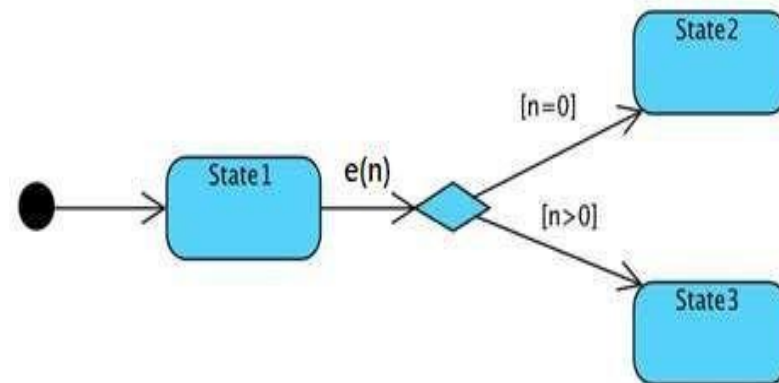
■ Unión (estática)



Las guardias

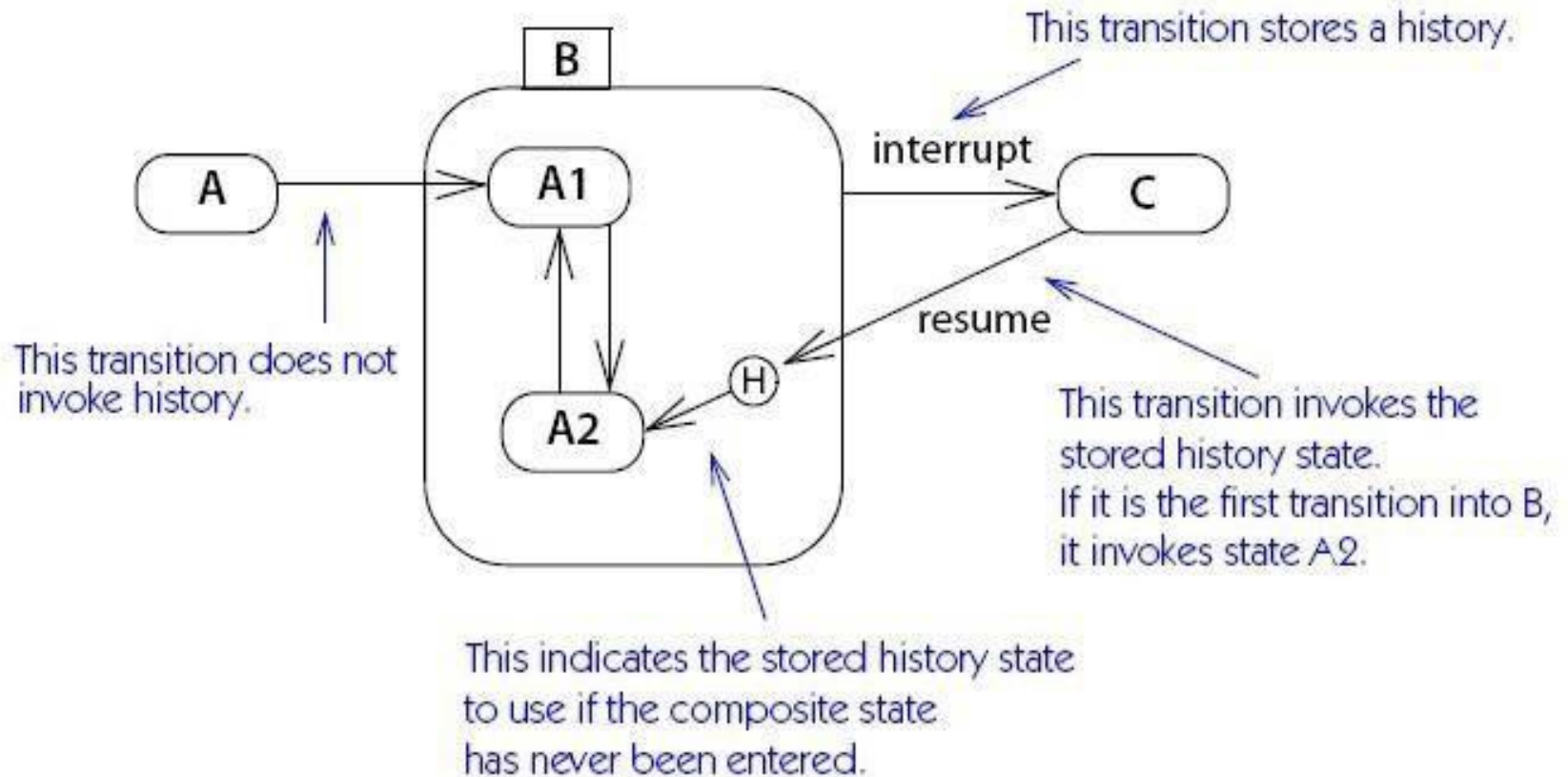
las guardias se evalúan antes de salir del Estado1. Si $n < 0$, el evento e se ignora y no se toma ninguna transición.

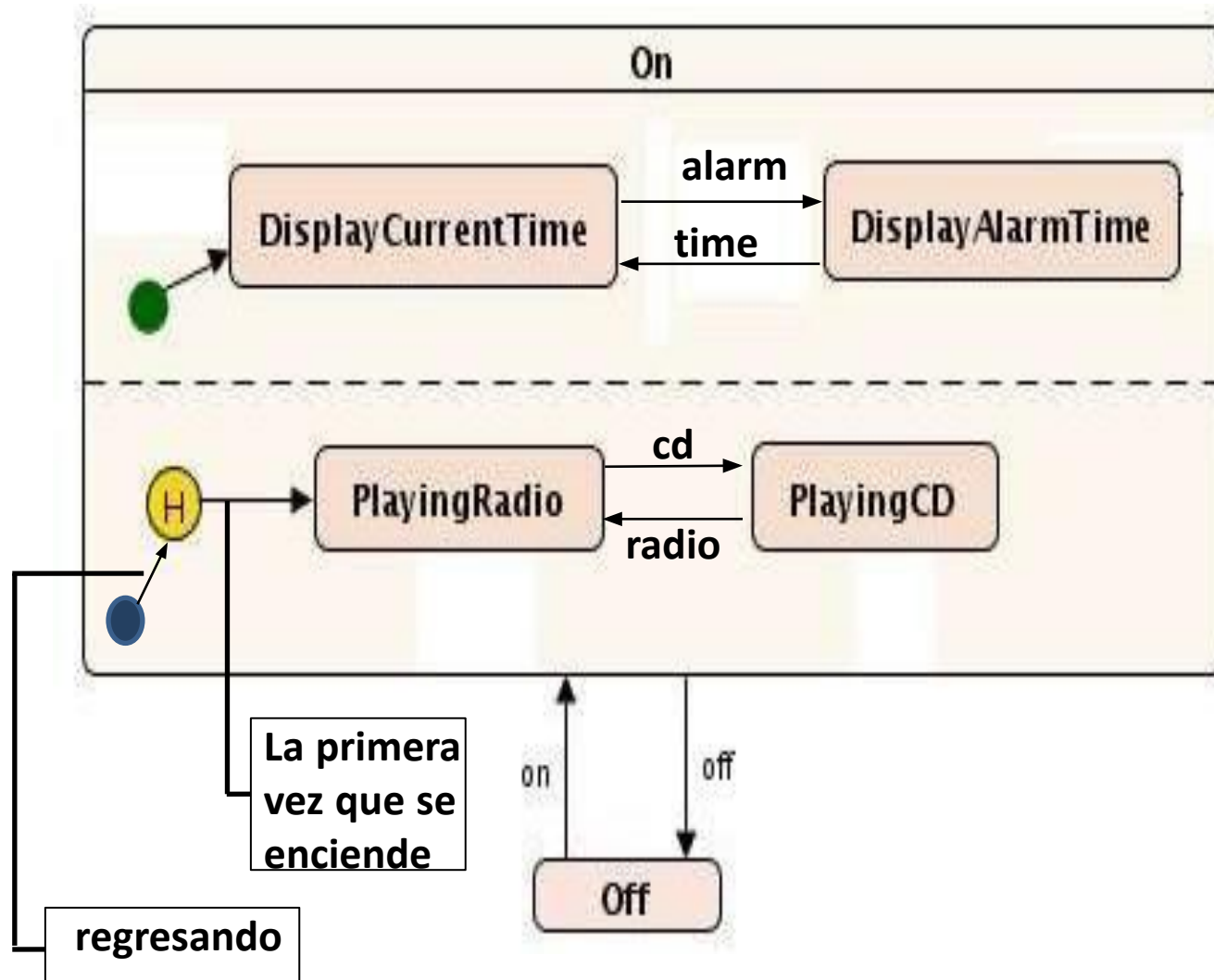
■ Choice (dinámica)



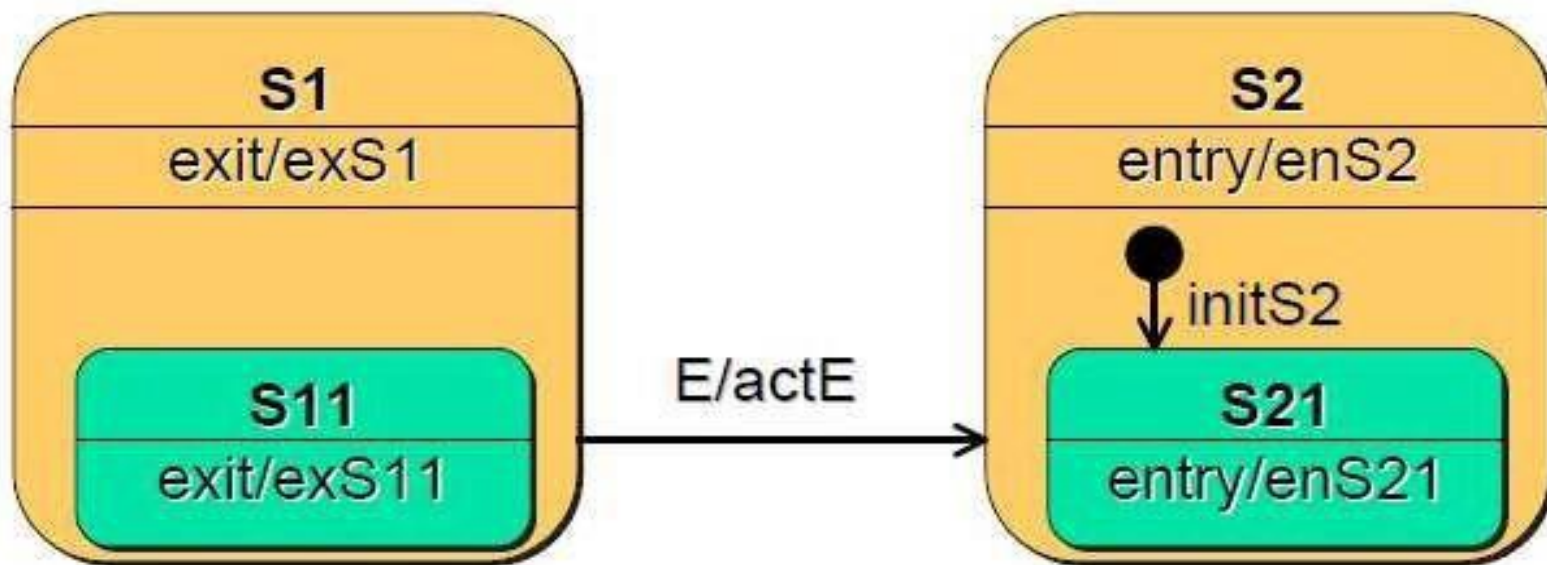
las guardias se evalúan después de e(n). En este ejemplo hay que garantizar que n es mayor o igual que cero

History state





Ejemplo



Actions execution sequence:

exS11 \Rightarrow exS1 \Rightarrow actE \Rightarrow enS2 \Rightarrow initS2 \Rightarrow enS21

Describir el modelo dinámico: ¿máquina a estados o actividades?

- Cómo elegir el diagrama más adecuado:
 - Si el foco es
 - poner en orden un conjunto de acciones a realizar → actividad
 - mostrar la evolución de un objeto en respuesta a eventos → estados
 - El diagrama de la máquina de estados habla de la evolución en el tiempo de las instancias de un clasificador
 - el diagrama de actividades forma parte de una agenda de acciones a realizar

Describir el modelo dinámico: nombres de estados y acciones

- Nombres de los estados:
 - adjetivos: activo,
 - participios pasados: encendida, apagada, pinned
 - gerundios: marcando, conectando
 - Otros: enEspera
- Nombres de las acciones:
 - verbos en indicativo, imperativo o infinitivo: crear, enviar
 - sustantivos que indican una acción: consulta DB
- No es una regla fija y a menudo se ignora en la práctica (también hay excepciones en los ejemplos vistos), pero seguir la regla es una buena manera de construir diagramas correctamente
 - Error común en tareas que confunden estados y acciones

UML REFERENCE MANUAL

Tipos de eventos

Table 7-1: *Kinds of Events*


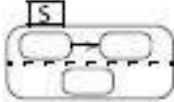
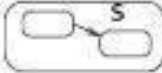






<i>Event Type</i>	<i>Description</i>	<i>Syntax</i>
call event	Receipt of an explicit synchronous call request by an object	op (a:T)
change event	A change in value of a Boolean expression	when (exp)
signal event	Receipt of an explicit, named, asynchronous communication among objects	sname (a:T)
time event	The arrival of an absolute time or the passage of a relative amount of time	after (time)

Tipos de transiciones

Table 7-2: Kinds of Transitions and Implicit Effects

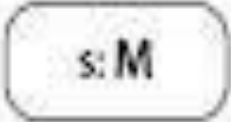


<i>Transition Kind</i>	<i>Description</i>	<i>Syntax</i>
entry transition	The specification of an entry activity that is executed when a state is entered	entry/ activity
exit transition	The specification of an exit activity that is executed when a state is exited	exit/ activity
external transition	A response to an event that causes a change of state or a self-transition, together with a specified effect . It may also cause the execution of exit and/ or entry activities for states that are exited or entered.	e(a:T)[guard]/activity
internal transition	A response to an event that causes the execution of an effect but does not cause a change of state or execution of exit or entry activities	e(a:T)[guard]/activity

Table 7-3: Kinds of States

State Kind	Description	Notation
simple state	A state with no substructure	
orthogonal state	A state that is divided into two or more regions. One direct substate from each region is concurrently active when the composite state is active.	
nonorthogonal state	A composite state that contains one or more direct substates, exactly one of which is active at one time when the composite state is active	
initial state	A pseudostate that indicates the starting state when the enclosing state is invoked	
final state	A special state whose activation indicates the enclosing state has completed activity	
terminate	A special state whose activation terminates execution of the object owning the state machine	
junction	A pseudostate that chains transition segments into a single run-to-completion transition	
choice	A pseudostate that performs a dynamic branch within a single run-to-completion transition	
history state	A pseudostate whose activation restores the previously active state within a composite state	

Tipos de estado

Submáquinas

<i>State Kind</i>	<i>Description</i>	<i>Notation</i>
submachine state	A state that references a state machine definition, which conceptually replaces the submachine state	
entry point	A externally visible pseudostate within a state machine that identifies an internal state as a target	
exit point	A externally visible pseudostate within a state machine that identifies an internal state as a source	

Una guía completa y completa de UML 2.5

← → ↻ 🔒 uml-diagrams.org



Google Custom Search 🔍

[Home](#) [UML Diagrams](#) [Class Diagrams](#) [Composite Structures](#) [Packages](#) [Components](#) [Deployments](#) [Use Case Diagrams](#) [Information Flows](#) [Activities](#)
[State Machines](#) [Sequence Diagrams](#) [Communications](#) [Timing Diagrams](#) [Interaction Overviews](#) [Profiles](#) [UML Index](#) [Examples](#) [About](#)

The Unified Modeling Language

The **Unified Modeling Language™** (UML®) is a standard visual modeling language intended to be used for

- modeling business and similar processes,
- analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

UML can be applied to diverse **application domains** (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component **software development methods** and for various **implementation platforms** (e.g., J2EE, .NET).

UML is a standard modeling **language**, not a **software development process**. [UML 1.4.2 Specification](#) explained that process:

- provides guidance as to the order of a team's activities,
- specifies what artifacts should be developed,
- directs the tasks of individual developers and the team as a whole, and
- offers criteria for monitoring and measuring a project's products and activities.

UML is intentionally **process independent** and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is **Rational Unified Process** (RUP).

Lo que se modele depende del destinatario y la perspectiva

Información

