

# 1. UML avanzado

## Índice

- Referencias
- Introducción
- Elementos de la notación
- Diagramas de casos de uso
  - Definición.
  - Notación.
  - Relaciones

## Índice

- Diagramas de actividades
  - Definición
  - Notación.
  - Nodos de llamada
  - Nodos adicionales
  - Flujos de control paralelos.
  - Calles (*swimlanes*).
  - Flujo de objetos.

## Índice

- Conectores
  - Aspectos avanzados
- Diagramas de clases
  - Definición.
  - Notación.

# Índice

- Diagramas de objetos
  - Definición.
  - Notación.
- Diagramas de interacción
  - Definición.
  - Diagramas de secuencia y comunicación.

# Índice

- Diagramas de secuencia
  - Definición.
  - Notación.
  - Restricciones de tiempo y localización
  - Diagramas de visión de conjunto de interacción

# Índice

- Diagramas de comunicación
  - Definición.
  - Notación.
- Diagramas de estados (*statecharts*)
  - Definición.
  - Notación.
  - Estados.
  - Transiciones.

# Índice

- Eventos.
- Condiciones de guarda.
- Otras características.
- Subestados.
- Diagramas de actividades y de estados
- Otros elementos
- Tipos de máquina
- Máquinas de estados de protocolo

# Índice

- Paquetes
  - Definición.
  - Notación.
  - Características.
  - Relaciones.
  - Estereotipos

# Índice

- Diagramas de componentes
  - Definición
  - Subsistemas
  - Puertos
- Colaboraciones
- Diagramas de despliegue
  - Definición
  - Notación
  - Arterfactos

# Índice

- Diagramas de estructura compuesta
  - Definición
  - Clases
  - Componente
- Diagramas de tiempo
  - Definición
  - Notación

# Índice

- Racionalidad de uso de UML
- Análisis y diseño OO
- Conclusiones

## Referencias

- Arlow, J., Neudstadt, I. *UML 2*. Anaya Multimedia, 2006
- Rumbaugh, J., Booch, G., Jacobson, I. *El lenguaje unificado de modelado. Manual de referencia*. Addison-Wesley, 2004
- Booch, G., Rumbaugh, J., Jacobson, I. *El lenguaje unificado de modelado. 2ª edición*. Addison-Wesley, 2006

## Referencias

- Eriksson, H.-E., Penker, M., Lyons, B., Fado, D. *UML 2 Toolkit*. Wiley Publishing Inc., 2004
- Booch G., *Análisis y diseño orientado a objetos con aplicaciones*, Segunda edición, Addison-Wesley/Díaz de Santos, 1996
- Jacobson I., Booch G., Rumbaugh J., *El proceso unificado de desarrollo de software*. Addison-Wesley 2000

## Introducción

- *Unified Modeling Language*, UML\*, es una notación que representa gráficamente construcciones del modelo de objetos
- La acción de dibujar un diagrama no constituye ni análisis ni diseño
- El diagrama se limita a capturar una descripción del comportamiento del sistema (análisis) o la visión y detalles de una arquitectura (diseño)

\*<http://www.uml.org/>

## Introducción

- Ventajas de una notación *expresiva y bien definida*:
  - Una notación *estándar* posibilita al analista o diseñador describir un universo o formular una arquitectura y comunicar estas decisiones de forma no ambigua.
  - Al aliviar al cerebro de todo trabajo innecesario, una buena notación lo libera para concentrarse en trabajos más avanzados.

## Introducción

- Una notación expresiva permite comprobar la consistencia y corrección de las decisiones adoptadas mediante herramientas automáticas.
- UML:
  - Notación Booch, Booch.
  - *Object Modeling Technique*, OMT, Rumbaugh et al.
  - *Object-Oriented Software Engineering*, OOSE, Jacobson.

## Elementos de la notación

- Como ya sabemos, no se pueden capturar todos los aspectos de un sistema complejo en una sola vista
- La *vista lógica* de un sistema sirve para describir la existencia y significado de las abstracciones principales y los mecanismos que forman el espacio del problema, o para definir la arquitectura del sistema

## Elementos de la notación

- La *vista física* de un sistema describe la composición concreta en cuanto a hardware y software del contexto o implantación del sistema
- La *vista estructural* de un sistema describe los aspectos estáticos, i.e., aquellos relativamente estables que caracterizan su estructura

## Elementos de la notación

- La *vista dinámica* de un sistema tiene en cuenta las partes mutables de éste, ligada principalmente a *eventos*

## Elementos de la notación

VISTAS	Estructural	Dinámica
<b>Lógica</b>	D. Casos Uso D. Clases D. Componentes D. Estructura Compuesta D. Objetos	D. Actividades D. Secuencia D. Comunicación D. Estados D. Tiempo
<b>Física</b>	D. Despliegue	

Vistas y modelos UML 2

## Elementos de la notación

- Durante este tema veremos las relaciones que existen entre los diversos diagramas y como se integran en un modelo de proceso
- Aunque en la *Guía del Usuario de UML* el orden de presentación de los diagramas es en base al tipo de información que modelan:

## Elementos de la notación

- Estructural:
  - Clases.
  - Objetos.
  - Componentes
- Comportamiento:
  - Casos de uso (desajuste en la segunda edición).
  - Actividades.
  - Interacción (secuencia y comunicación).
  - Estados.

## Elementos de la notación

- Arquitectónico:
  - Despliegue.
- Nosotros los veremos en un orden más centrado en el orden de uso de este tipo de diagramas:
  - Casos de uso.
  - Actividades.
  - Clases.

## Elementos de la notación

- Objetos.
- Interacción:
  - Secuencia.
  - Comunicación.
- Estados.
- Componentes.
- Despliegue.
- Estructura compuesta.
- Tiempo

## Diagramas de casos de uso Definición

- Un *diagrama de casos de uso* es un diagrama que muestra un conjunto de casos de uso, actores y sus relaciones
- Un *caso de uso* es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor

## Diagramas de casos de uso Definición

- Un *actor* representa un conjunto coherente de roles que los usuarios de los casos de uso juegan al interactuar con estos.  
Normalmente un actor es un rol que es jugado por una persona, un dispositivo hardware o incluso otro sistema al interactuar con nuestro sistema

## Diagramas de casos de uso Notación

- Los elementos más comunes de los diagramas de casos de uso son:


- Caso de uso: 

- Actor: 

- Relación:

- Generalización: 

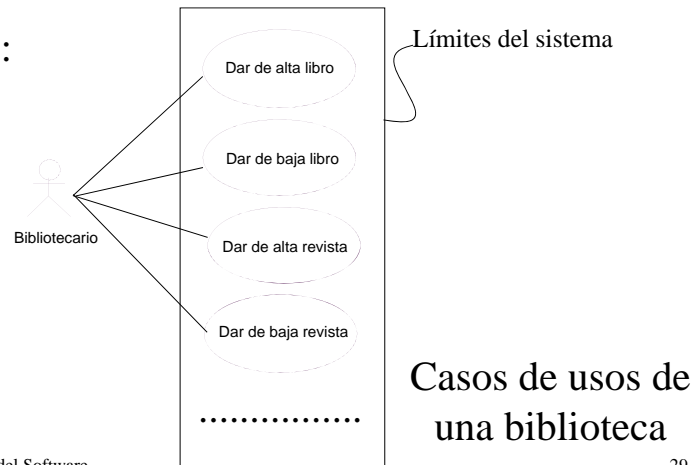
- Inclusión: 

- Extensión: 

## Diagramas de casos de uso

### Notación

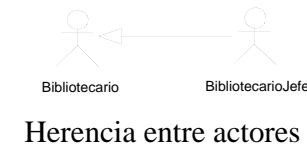
- Ej.:



## Diagramas de casos de uso

### Relaciones

- Las relaciones entre los actores pueden ser de generalización: el actor hijo puede ser utilizado en cualquier caso de uso donde se utilice el actor padre
- Ej.:



## Diagramas de casos de uso

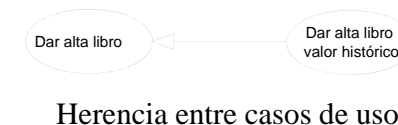
### Relaciones

- Las relaciones entre casos de uso pueden ser de:
  - Generalización.
  - Inclusión.
  - Extensión.
- Generalización
  - El caso hijo hereda el comportamiento y el significado del caso de uso padre.

## Diagramas de casos de uso

### Relaciones

- El hijo puede añadir o redefinir el comportamiento del padre.
- El hijo puede ser colocado donde aparezca el padre.
- Ej.:



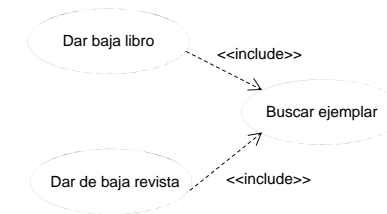


## Diagramas de casos de uso Relaciones

- Inclusión
  - El caso de uso base incorpora explícitamente el comportamiento de otro caso de uso especificado en el caso base.
  - El caso de uso incluido nunca se encuentra aislado, sino que es instanciado sólo como parte de algún caso base más amplio que lo incluya.

## Diagramas de casos de uso Relaciones

- Esta relación se utiliza para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte, que será incluido por el caso de uso base.
- Ej.:



## Diagramas de casos de uso Relaciones

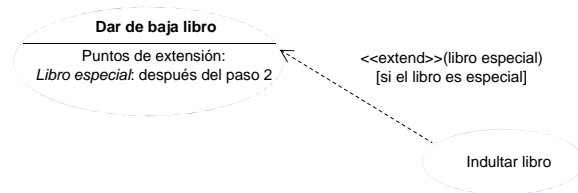
- Dar de baja libro:
  - Flujo de eventos principal:
    1. Obtener datos.
    2. Buscar ejemplar.
    3. Si se encuentra y no está prestado: eliminar libro.
    4. Notificar usuario.
  - Flujo de eventos alternativo:
    - Si en 3. no se encuentra o está prestado no eliminar y notificar al usuario.

## Diagramas de casos de uso Relaciones

- Extensión
  - Una relación de extensión entre casos de uso se utiliza para extender condicionalmente el comportamiento de un caso de uso existente.
  - Es una forma de añadir comportamiento al caso base sin cambiarlo.
  - Normalmente se utiliza:
    - Cuando se trabaja en versiones posteriores.
    - Para indicar lugares donde se puede personalizar.

## Diagramas de casos de uso Relaciones

- Ej.:



Extensión entre casos de uso

## Diagramas de casos de uso Relaciones

- Dar de baja libro:

- Flujo de eventos principal:

1. Obtener datos.
2. Buscar ejemplar.
3. Si se encuentra y no está prestado: eliminar libro.
4. Notificar usuario.

- Flujo de eventos alternativo:

- Si en 3. no se encuentra o está prestado no eliminar y notificar al usuario.

- Indultar libro (después del paso 2):

- Flujo de eventos principal:

1. Comunicar que el libro no se da de baja por ser especial.

## Diagramas de actividades Definición

- Los diagramas de actividades muestran un flujo de actividades
- Están formados por *nodos* y *arcos*
- En UML 1.x se utilizaba la nomenclatura de *estados* y *transiciones*




## Diagramas de actividades Definición

- Los nodos pueden ser:
  - De acción: unidades atómicas de trabajo dentro de la actividad
  - De control: controlan el flujo en la actividad
  - De objetos: objetos usados en la actividad

## Diagramas de actividades

### Notación

- Los elementos más comunes de los diagramas de actividades son:

- Nodo: 
- Arco: 
- Estado inicial: ●
- Estado final: ●
- Bifurcación: 

## Diagramas de actividades

### Notación

- Ej. Dar de baja libro:

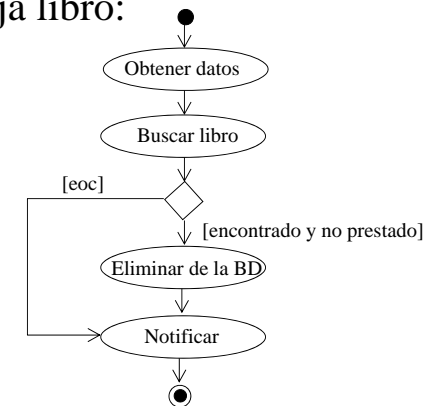


Diagrama de actividades

## Diagramas de actividades

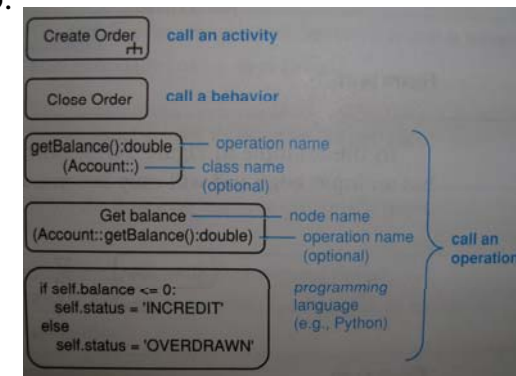
### Nodos de llamada

- Dentro de los nodos de acción encontramos los de *llamada*, que pueden invocar:
  - Una actividad
  - Un comportamiento
    - En otra actividad
    - En otra máquina de estados
    - En otro diagrama de interacción
  - Una operación

## Diagramas de actividades

### Nodos de llamada

- Ejemplo:



Nodos de llamada

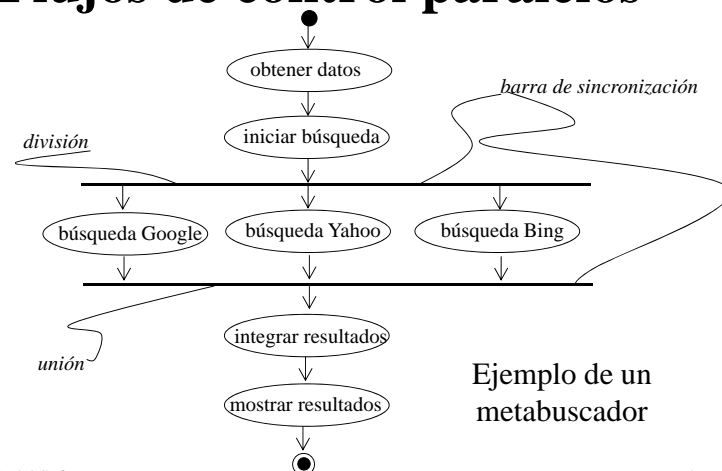
## Diagramas de actividades Nodos adicionales

- Además, UML 2.x considera que puede haber nodos de control adicionales:
  - De final de flujo: ⊗
  - De fusión de flujo: ◇

## Diagramas de actividades Flujos de control paralelos

- Podemos encontrarnos con *flujos concurrentes*, los cuales lanzan nodos que se ejecutan en paralelo
- En UML se utiliza una *barra de sincronización* para especificar la división y unión de estos flujos de control paralelos
- Ej.:

## Diagramas de actividades Flujos de control paralelos



Ejemplo de un  
metabuscador

## Diagramas de actividades Flujos de control paralelos

- Una barra de sincronización de *división* representa la separación de un flujo de control en dos o más flujos concurrentes
- Una división tiene:
  - Al menos un arco de entrada.
  - Dos o más arcos de salida (flujos independientes).

## Diagramas de actividades

### Flujos de control paralelos

- Después de la división, las actividades asociadas a cada uno de estos caminos continúan en paralelo.
- Una barra de sincronización de *unión* representa la sincronización de dos o más flujos concurrentes

## Diagramas de actividades

### Flujos de control paralelos

- Una unión tiene:
  - Al menos dos arcos de entrada.
  - Un arco de salida.

## Diagramas de actividades

### Flujos de control paralelos

- Antes de llegar a la unión, las actividades asociadas con cada camino continúan en paralelo.
- En la unión, los flujos concurrentes se sincronizan, es decir, cada uno espera hasta que todos los flujos de entrada han alcanzado la unión.

## Diagramas de actividades

### Flujos de control paralelos

- A partir de ahí, continúa un único flujo de control que sale de la unión.
- Por último, las uniones y divisiones deben *equilibrarse*: el número de flujos que parten de una división debe coincidir con el número de flujos que entran en la unión correspondiente

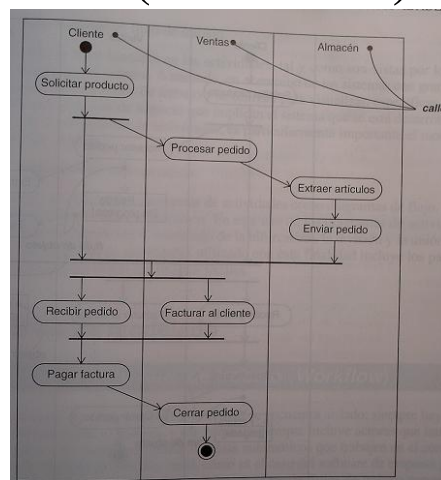
## Diagramas de actividades Calles (*swimlanes*)

- Un diagrama de actividades puede contener grupos de actividades relacionadas
- Pueden agruparse en base a:
  - Los objetos o actores que llevan a cabo las actividades.
  - Los casos de uso de los que forman parte.
  - Otro criterio.

## Diagramas de actividades Calles (*swimlanes*)

- Para ilustrar este hecho se divide el diagrama de actividades en particiones denominadas *calles*
- Ej.:

## Diagramas de actividades Calles (*swimlanes*)



Calles

## Diagramas de actividades Flujo de objetos

- En el flujo de control de un diagrama de actividades pueden verse involucrados objetos
- Esto se debe a que las actividades manipulan objetos

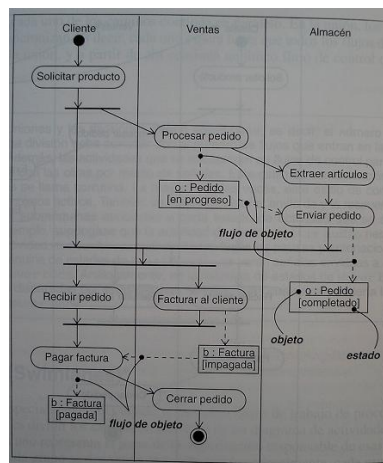
## Diagramas de actividades Flujo de objetos

- Se pueden especificar los objetos implicados en un diagrama de actividades conectándolos con una dependencia a la actividad o transición que los crea, los destruye o modifica

## Diagramas de actividades Flujo de objetos

- Este uso de las relaciones de dependencia y de los objetos se denomina *flujo de objetos* porque representa la participación de un objeto en un flujo de control
- Además del flujo de objetos, es posible mostrar cómo cambian los valores de sus atributos, su estado y sus roles

## Diagramas de actividades Flujo de objetos



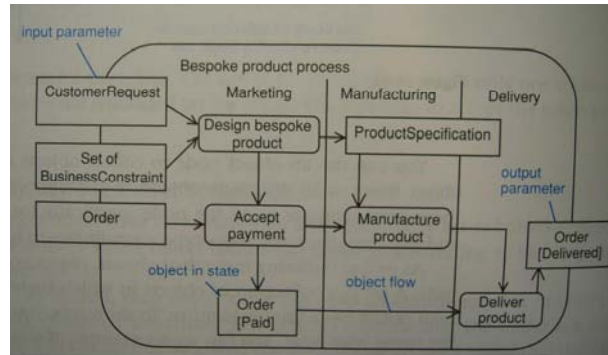
Flujo de objetos

## Diagramas de actividades Flujo de objetos

- También podemos especificar parámetros de entrada y de salida

## Diagramas de actividades Flujo de objetos

- Ejemplo:



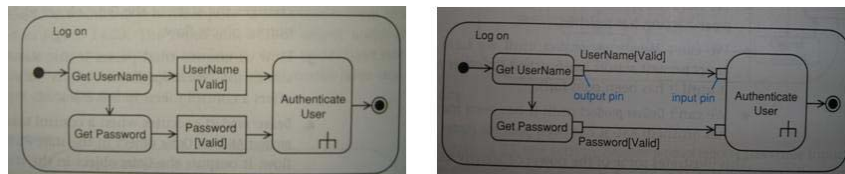
Flujo de objetos y parámetro de actividad

## Diagramas de actividades Conectores

- Los nodos de llamada pueden incluir *conectores (pin)*
- Estos conectores son nodos que representan entradas o salidas de acciones
- Así, por ejemplo, si los diagramas de actividades incluyen muchos objetos de entrada y salida, se pueden utilizar en su lugar conectores de entrada y salida

## Diagramas de actividades Conectores

- Ejemplo



Diagramas equivalentes sin y con conectores

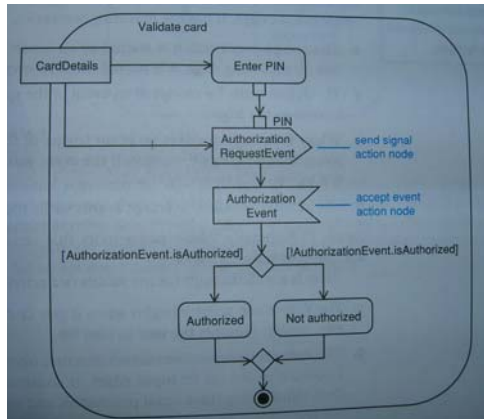
## Diagramas de actividades Aspectos avanzados

- Otros nodos de acción:
  - Nodo de envío de señal (asíncrona):
  - Nodo de aceptación de un evento:
  - Nodo de aceptación de evento de tiempo:



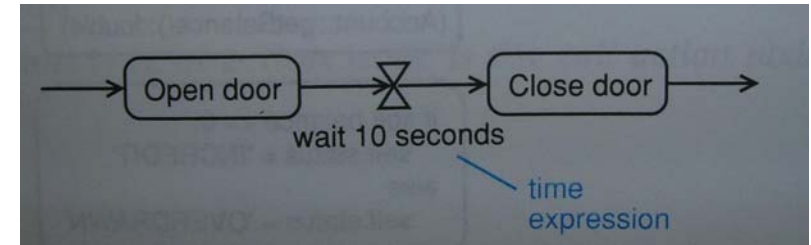
## Diagramas de actividades

### Aspectos avanzados



## Diagramas de actividades

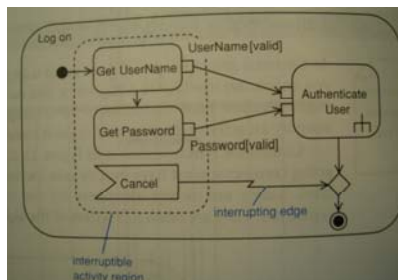
### Aspectos avanzados



## Diagramas de actividades

### Aspectos avanzados

- Las *regiones de actividad interrumpibles* denotan regiones de actividad que se interrumpen al activar un arco de interrupción
- Ejemplo:



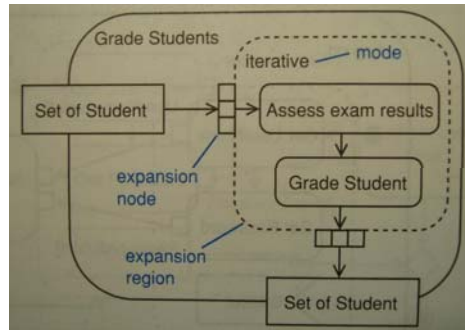
## Diagramas de actividades

### Aspectos avanzados

- Los *nodos de expansión* permiten especificar el procesamiento de una colección de objetos por una parte del diagrama de actividades denominada *región de expansión*
- El procesamiento puede ser:
  - iterative: iterativo
  - parallel: paralelo
  - stream: procesa cada elemento según llega

## Diagramas de actividades Aspectos avanzados

- Ejemplo:



Nodo de expansión

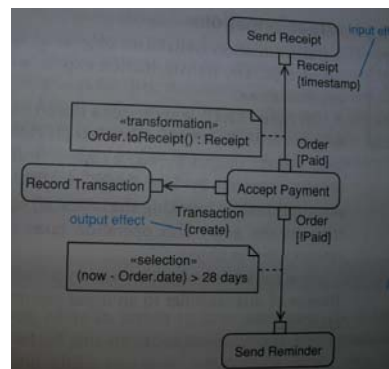
## Diagramas de actividades Aspectos avanzados

- Otros

- Los *efectos de entrada y salida* muestran los efectos de una acción sobre un objeto
- Una *selección* es una condición impuesta a un flujo de objetos que le lleva a aceptar únicamente aquellos objetos que satisfacen la condición
- Una *transformación* transforma objetos en un flujo de objetos a objetos de diferente tipo

## Diagramas de actividades Aspectos avanzados

- Ejemplo:

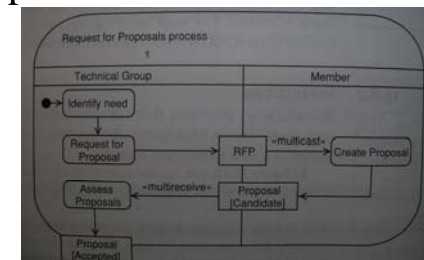


Ejemplo de efectos, selecciones y transformaciones

## Diagramas de actividades Aspectos avanzados

- El *multienvío* y la *multirecepción* permite enviar y recibir más de un objeto a más de un emisor y receptor

- Ejemplo

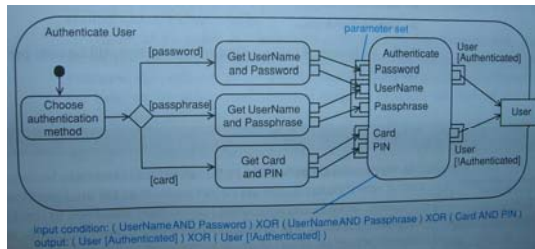


Ejemplo de multienvío y multirecepción

## Diagramas de actividades

### Aspectos avanzados

- Los *conjuntos de parámetros* definen conjuntos alternativos de conectores de entrada y salida
- Ejemplo:



Ingeniería del Software  
Antonio Navarro

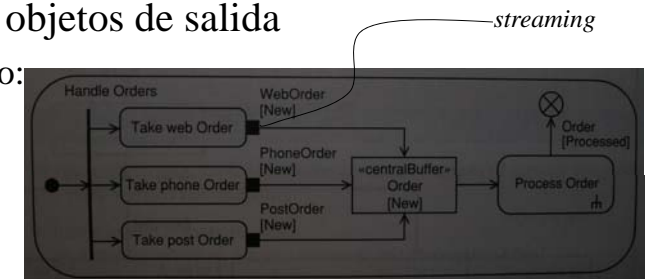
Conjuntos de parámetros

73

## Diagramas de actividades

### Aspectos avanzados

- Un *almacen central* permite combinar múltiples flujos de objetos de entrada y distribuir dichos objetos entre múltiples flujos de objetos de salida
- Ejemplo:



Ingeniería del Software  
Antonio Navarro

Almacén central

74

## Diagramas de clases

### Definición

- Un *diagrama de clases* es un diagrama que muestra un conjunto de clases, interfaces y sus relaciones

Ingeniería del Software  
Antonio Navarro

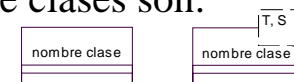
75

## Diagramas de clases

### Notación

- Los elementos más comunes de los diagramas de clases son:

– Clases:



– Interfaces:



– Relaciones:

- Generalización:
- Asociación:
- Agregación:

Ingeniería del Software  
Antonio Navarro

76

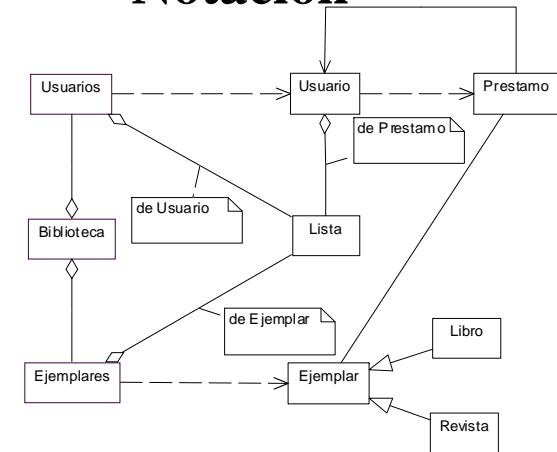
## Diagramas de clases

### Notación

- Dependencia: .....>
  - Instanciación: .....>
  - Metaclass: .....>
  - Realización: .....>
- Ej.:

## Diagramas de clases

### Notación



## Diagramas de objetos

### Definición

- Un *diagrama de objetos* es un diagrama que representa un conjunto de objetos y sus relaciones en un momento concreto.
- Estos diagramas modelan instancias de los elementos contenidos en los diagramas de clases

## Diagramas de objetos

### Notación

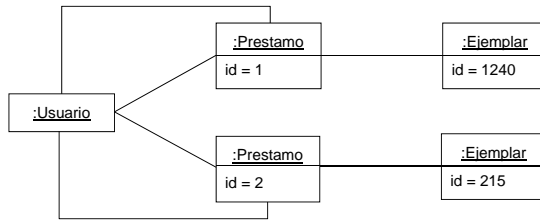
- Los elementos más comunes de los diagramas de objetos son:
  - Objeto: 

<u>objeto:Clase</u>
atributo <sub>1</sub> = valor1
.....
atributo <sub>k</sub> = valor <sub>k</sub>
  - Enlace: \_\_\_\_\_

## Diagramas de objetos

### Notación

- Ejemplo



Modelado de estructuras de objetos

## Diagramas de interacción

### Definición

- Los *diagramas de interacción* muestran una interacción, la cual consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos
- Una *interacción* es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos dentro de un contexto para lograr un propósito

## Diagramas de interacción

### Definición

- Un *mensaje* es la especificación de una comunicación entre objetos que transmiten información con expectativa de desencadenar una actividad
- Un *contexto* es un conjunto de objetos que están relacionados para un propósito

## Diagramas de interacción

### Definición

- A nivel contextual, la operación puede interpretarse:
  - A nivel sistema.
  - A nivel implementación de operaciones.
  - A nivel clase
- Discusión: a nivel código, ¿dónde se dan las interacciones?

## Diagramas de interacción

### Definición

- Los elementos más comunes de los diagramas de interacción son:
  - Objetos.
  - Enlaces.
  - Mensajes.

## Diagramas de interacción

### Diagramas de secuencia...

- Hay dos tipos de diagramas de interacción:
  - Diagramas de secuencia.
  - Diagramas de comunicación.
- Los diagramas de secuencia y de comunicación expresan una información similar, pero la muestran de maneras diferentes

## Diagramas de interacción

### Diagramas de secuencia...

- Los diagramas de secuencia muestran la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos.
- Los diagramas de comunicación muestran las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto y para el diseño de procedimientos

## Diagramas de secuencia

### Definición

- Un *diagrama de secuencia* muestra las interacciones entre objetos organizadas en una secuencia temporal
- En particular, muestra los objetos participantes en la interacción y la secuencia de mensajes intercambiados

## Diagramas de secuencia

### Definición

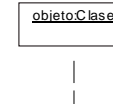
- Un *mensaje* denota el hecho de aportar información de un objeto (u otra instancia) a otro, con esperanzas de que esto dé lugar a alguna actividad
- Los diagramas de secuencia muestran la traza de ejecución de un escenario, destacando la ordenación temporal de los mensajes

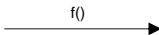
## Diagramas de secuencia


### Notación

- Los elementos más comunes de los diagramas de secuencia son:

– Objeto:



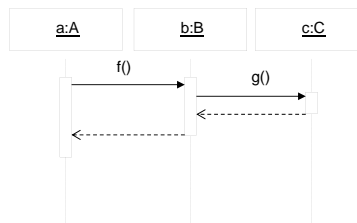
– Llamada u otro flujo anidado: 

– Retorno: 

– Asíncrono, sin anidamiento de control: 

## Diagramas de secuencia

### Notación



Llamada a procedimientos  
en un flujo de control anidado

## Diagramas de secuencia

### Restricciones de tiempo y local.

- Para representar las necesidades de modelado de los sistemas de tiempo real y distribuidos, UML proporciona una representación gráfica para las marcas de tiempo, las expresiones de tiempo y las restricciones de tiempo y localización

## Diagramas de secuencia

### Restricciones de tiempo y local.

- Una *marca de tiempo* denota el instante en el que ocurre un evento
- Una *expresión de tiempo* al evaluarse genera un valor de tiempo absoluto o relativo, o puede ser el nombre de un mensaje
- Una *restricción de tiempo* es un enunciado semántico sobre el valor absoluto o relativo del tiempo

## Diagramas de secuencia

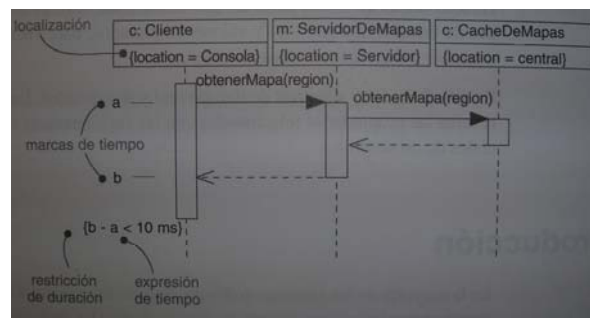
### Restricciones de tiempo y local.

- Una *localización* es la asignación de un componente a un nodo de ejecución

## Diagramas de secuencia

### Restricciones de tiempo y local.

- Ejemplo:



Restricciones de tiempo y localización

## Diagramas de secuencia

### Diagramas de visión de ...

- Los *diagramas de visión de conjunto de interacción* son una forma especial de diagrama de actividad que modelan el flujo de control de alto nivel entre interacciones
- Los nodos son interacciones o referencias a interacciones
- Son muy útiles para mostrar el flujo de control entre casos de uso



## Diagramas de secuencia

## Diagramas de visión de ...

- Ejemplo:

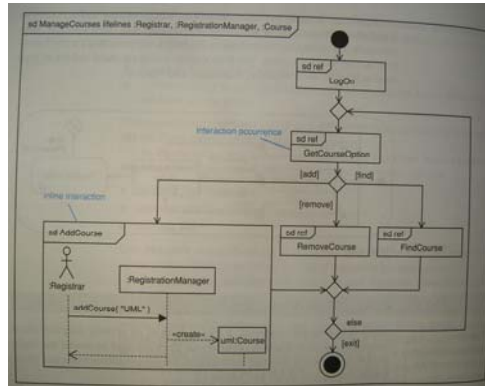


Diagrama de visión de conjunto de interacción

Ingeniería del Software  
Antonio Navarro

97

## Diagramas de comunicación

## Definición

- Un *diagrama de comunicación* muestra las interacciones entre objetos destacando su organización
- En UML 1.x se denominaban de *colaboración*
- Respecto a los diagramas de secuencia:
  - Permiten indicar la visibilidad del enlace.
  - Es necesario indicar explícitamente el número de secuencia

Ingeniería del Software  
Antonio Navarro

98

## Diagramas de comunicación

## Definición

- Son semánticamente equivalentes
- Esto no evita que unos permitan especificar detalles que otros no pueden
  - Información de visibilidad en diagramas de comunicación.
  - Información de retorno en diagramas de secuencia.

Ingeniería del Software  
Antonio Navarro

99

## Diagramas de comunicación

## Definición

- Aunque visualmente son similares a los diagramas de objetos tienen significados distintos:
  - D. objetos: foto del sistema / objeto prototípico.
  - D. comunicación: interacciones entre los objetos del sistema para llevar a cabo alguna funcionalidad.

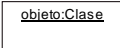
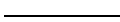


Ingeniería del Software  
Antonio Navarro

100

## Diagramas de comunicación

### Notación

- Los elementos más comunes de los diagramas de secuencia son:

- Objeto: 
- Enlace: 
- Llamada: 
- Envío: 

## Diagramas de comunicación

### Notación

- Ejemplo:

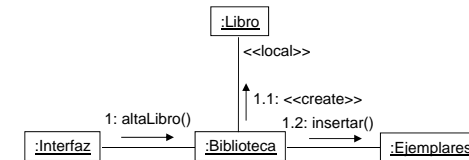


Diagrama de comunicación equivalente al diagrama de interacción de la t188

## Diagramas de estados

### Definición

- Un *diagrama de estados* muestra una máquina de estados, destacando el flujo de control entre estados
- Una *máquina de estados* es una especificación de las secuencias de estados por las que pasa un objeto a lo largo de su vida en respuesta a eventos, junto con las respuestas a esos eventos

## Diagramas de estados

### Definición

- Un *estado* es una condición o situación en la vida de un objeto durante la cual satisface una condición, realiza alguna actividad, o espera algún evento
- Un *evento* es la especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y en el espacio

## Diagramas de estados

### Definición

- En el contexto de las máquinas de estados, un *evento* es la aparición de un estímulo que puede activar una transición de estado
- 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 específicas

## Diagramas de estados

### Definición

- Una *actividad* es una ejecución no atómica en curso dentro de una máquina de estados
- Una *acción* es una computación atómica ejecutable que produce un cambio de estado del modelo o la devolución de algún valor

## Diagramas de estados

### Definición

- Los diagramas de estado representan el comportamiento de una clase o de todo el sistema
- Suelen utilizarse para modelar:
  - *Objetos reactivos* (o dirigidos por eventos). La mejor forma de caracterizar su comportamiento es señalar su respuesta los eventos lanzados desde fuera de su contexto.

## Diagramas de estados

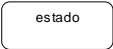
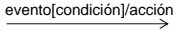
### Definición

- En general, objetos cuyo comportamiento dependa de su *pasado*, es decir, su comportamiento varía significativamente en función de su *estado* actual.

## Diagramas de estados

### Notación

- Los elementos más comunes de los diagramas de estados son:

- Estado: 
- Estado inicial: ●
- Transición: 

## Diagramas de estados

### Notación

- Ejemplo

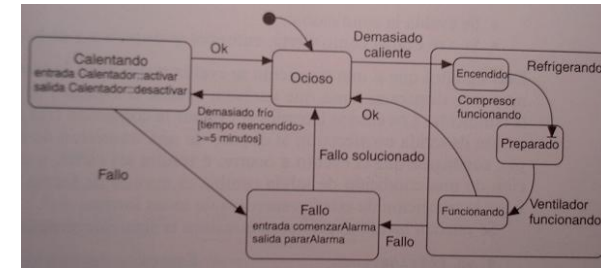


Diagrama de estados para un sistema de refrigeración/calefacción

## Diagramas de estados

### Notación

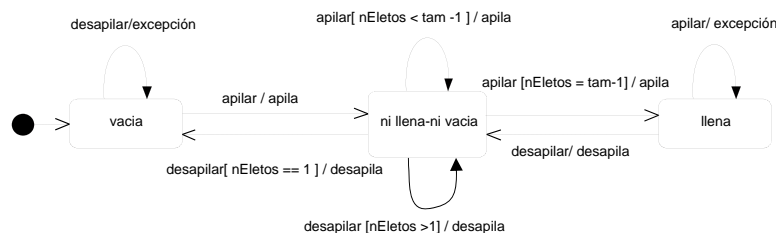


Diagrama de estados para una pila estática

## Diagramas de estados

### Estados

- Un *estado* es una condición o situación en la vida de un objeto durante la cual satisface alguna condición (e.g. vacía), realiza alguna actividad (e.g. calentando), o espera algún evento (e.g. ocioso)
- Un objeto permanece en un estado una cantidad de tiempo finita

## Diagramas de estados UML 1.x

### Estados

- Un diagrama de estados captura todos los comportamientos *posibles* de los objetos de una clase, pero no todos los objetos de la clase tienen que pasar por todos los estados (e.g. no todas las pilas tienen que llenarse)

## Diagramas de estados

### Estados

- En la máquina de estados de un objeto se pueden definir dos estados especiales
  - El *estado inicial*, que indica el punto de comienzo por defecto para la máquina de estados o el subestado. ●
  - El *estado final*, que indica que la ejecución de la máquina de estados, o del estado que lo contiene, ha finalizado. ◎

## Diagramas de estados

### Estados

- Ejemplo:

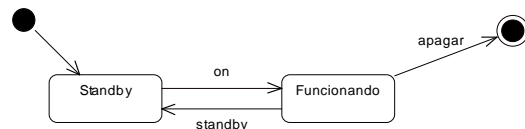


Diagrama de estados para un dispositivo electrónico

## Diagramas 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 específicas.
- Cuando se produce este cambio de estado decimos que la transición se ha *disparado/activado*

## Diagramas de estados

### Transiciones

- Una transición tiene cinco partes:
- *Estado origen*. Estado afectado por la transición. Si un objeto está en el estado origen, una transición de salida puede dispararse cuando el objeto reciba el evento de disparo de la transición, si satisface la condición de guarda (supuesto que haya).

## Diagramas de estados

### Transiciones

- *Evento de disparo*. Evento cuya recepción por el objeto que está en el estado origen provoca el disparo de la transición si se satisface su condición de guarda (supuesto que haya).
- Si no aparece evento de disparo, la transición se dispara implícitamente cuando el estado origen completa su actividad.

## Diagramas de estados

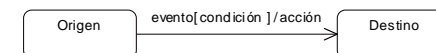
### Transiciones

- *Condición de guarda*. Expresión booleana que se evalúa cuando la transición se activa por la recepción del evento de disparo. Si la condición tiene el valor *verdadero*, la transición se dispara; si es *falso*, la transición no se dispara, y si no hay otra transición que pueda ser disparada, el evento se pierde.

## Diagramas de estados

### Transiciones

- *Acción*. Computación atómica ejecutable que puede actuar directamente sobre el objeto asociado a la máquina de estados, e indirectamente sobre otros objetos visibles al objeto.
- *Estado destino*. El estado activo tras completarse la transición.



Transición entre estados

## Diagramas de estados

### Eventos

- Un *evento* es la especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y el espacio
- En el contexto de las máquinas de estados, un evento es la aparición de un estímulo que puede disparar una transición de estado

## Diagramas de estados

### Eventos

- Los eventos pueden ser:
  - *Internos*. Son aquellos que fluyen entre los objetos de un sistema (e.g. la invocación de una función miembro).
  - *Externos*. Son aquellos que fluyen entre el sistema y sus actores (e.g. la pulsación de un botón).

## Diagramas de estados

### Eventos

- En UML, en el contexto de máquinas de estados, se pueden modelar cuatro tipos de eventos:
  - Señales.
  - Llamadas.
  - Paso del tiempo.
  - Cambio de estado.

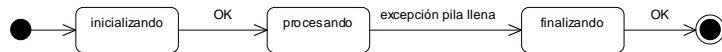
## Diagramas de estados

### Eventos

- Un *evento de señal* representa la recepción de una señal
- Una *señal* es una especificación de una comunicación asíncrona entre objetos
- Nótese que una señal puede aparecer en:
  - El evento de una transición en una máquina de estados.
  - La acción de una transición en una máquina de estados.

## Diagramas de estados Eventos

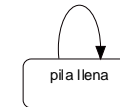
- El envío de un mensaje en una interacción.
- La ejecución de una operación también puede enviar señales.



Una señal como evento en una máquina de estados

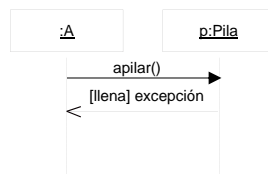
## Diagramas de estados Eventos

apilar / lanzar excepción



Una señal lanzada por una pila como acción en una máquina de estados

## Diagramas de estados Eventos



Una señal lanzada por una pila como un mensaje en un diagrama de interacción

## Diagramas de estados Eventos



Señales que puede enviar la función apilar() en un diagrama de clases



## Diagramas de estados Eventos

- Un *evento de llamada* representa la recepción de una petición para la invocación síncrona de una operación específica
- El resultado esperado es la ejecución de una secuencia de acciones que caracterizan el comportamiento de la operación en un estado particular

## Diagramas de estados Eventos

- Ejemplo:

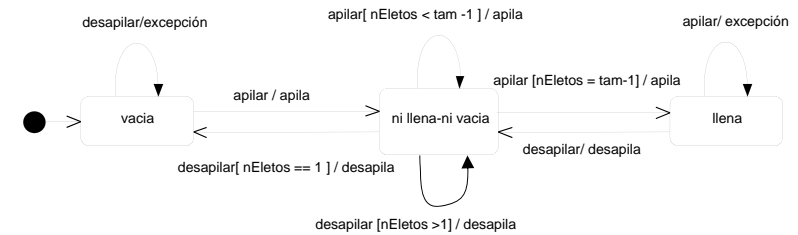
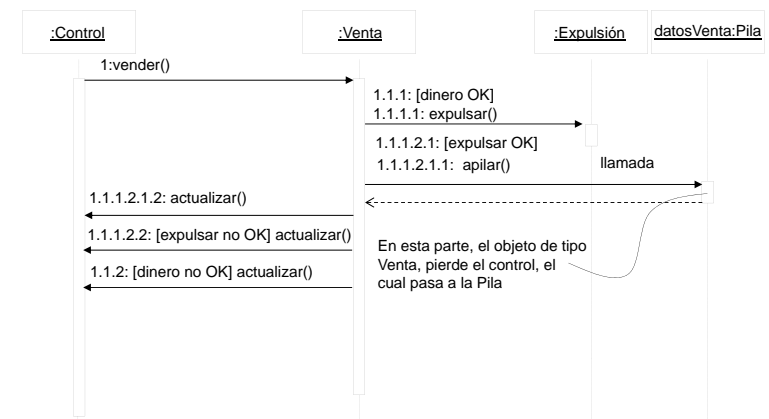


Diagrama de estados para una pila estática

## Diagramas de estados Eventos

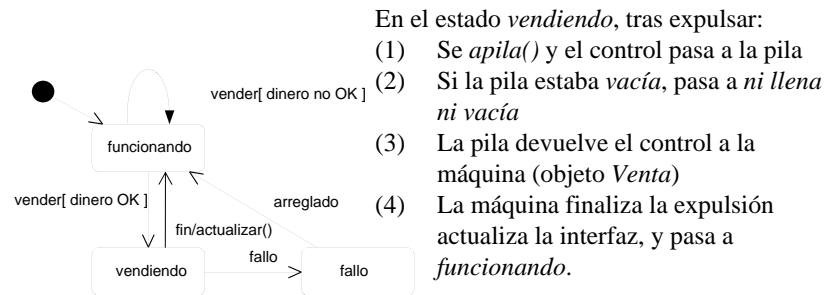
- Un evento de llamada significa que cuando un objeto invoca una operación sobre otro objeto que tiene una máquina de estados:
  - El control pasa del emisor al receptor.
  - El evento dispara la transición.
  - La operación acaba.
  - El receptor pasa a un nuevo estado.
  - El control regresa al emisor

## Diagramas de estados Eventos



Interacción de una venta en una máquina

## Diagramas de estados Eventos



Parte de la máquina de estados de una máquina de venta de un producto

## Diagramas de estados Eventos

- Nótese que los eventos de llamada (e.g. *apilar()*) deben reflejarse como una operación correspondiente en la clase (e.g. *Pila*)

## Diagramas de estados Eventos

- Un *evento de tiempo* es un evento que representa el paso del tiempo
  - Se modela con la palabra *after* seguida de alguna expresión que se evalúa para producir algún periodo de tiempo.
  - A menos que se especifique explícitamente, el periodo de tiempo asociado a una expresión de este tipo comienza en el instante a partir del cual se entra en el estado actual.

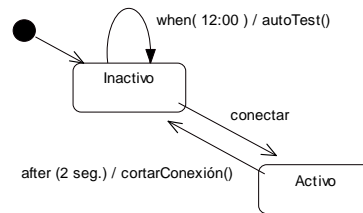
## Diagramas de estados Eventos

- Un *evento de cambio* es un evento que representa un cambio en el estado o el cumplimiento de alguna condición
  - Se modela con la palabra *when* seguida de alguna expresión booleana.
  - Se pueden utilizar expresiones para marcar un tiempo absoluto (*when hora == 12:00*) o para la evaluación continua de una expresión (*when temperatura < 15*).

## Diagramas de estados

### Eventos

- Ejemplo:



Eventos de tiempo y cambio

## Diagramas de estados

### Otras características

- Las *acciones de entrada y salida* se utilizan cuando se desea realizar una misma acción cada vez que se entra/sale de un estado sin importar la transición que nos lleva/saca a ese estado
- Las *transiciones internas* sirven para manejar eventos que no provocan un abandono del estado

## Diagramas de estados

### Otras características

- Difieren de las autotransiciones en que no se ejecutan las acciones de salida ni de entrada.
- Un estado puede representar la realización de una *actividad en curso*
  - Mientras se permanece en ese estado se realizará una tarea hasta que sea interrumpido por un evento.

## Diagramas de estados

### Otras características

- La actividad `do` podría:
  - Hacer referencia a otra máquina de estados.
  - Especificar una secuencia de acciones:  
`do op1 (a) ; op2 (b) ; op3 ( ) ;`  
 Las acciones no se pueden interrumpir, pero la secuencia de acciones sí.
- Los *eventos diferidos* son eventos a los que se proporcionará una respuesta en otro momento

## Diagramas de estados

### Otras características

- Un evento diferido es una lista de eventos cuya aparición en el estado se pospone hasta que se activa un estado en el que los eventos listados no han sido diferidos y pueden ocurrir, pudiendo disparar transiciones como si realmente acabaran de producirse.

## Diagramas de estados

### Otras características

- Ejemplo:

```
Rastreando
entry: activarModo(enRastreo)
exit: activarModo(noRastreo)
nuevoObjetivo / rastreador.adquirir()
do / seguirObjetivo
autoTest / defer
```

Estado *Rastreando*

## Diagramas de estados

### Subestados

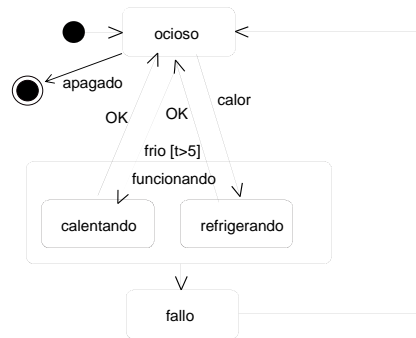
- Un *subestado* es un estado anidado dentro de otro
- Un estado con subestados se denomina *estado compuesto*
- Los estados compuestos surgen:
  - Como resultado de caracterizar una respuesta común a un evento por parte de diversos estados.

## Diagramas de estados

### Subestados

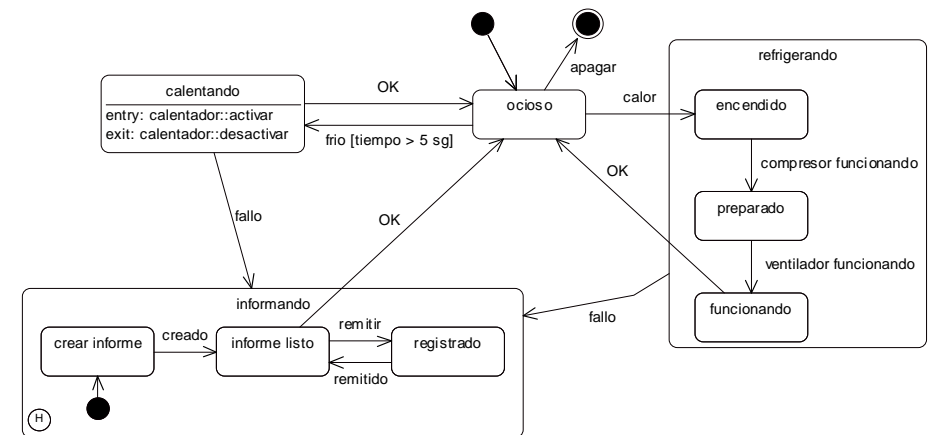
- Para describir con mayor nivel de detalle la actividad que se lleva a cabo dentro de un estado.
- Ejemplo:

## Diagramas de estados Subestados



Sistema de control de temperatura

## Diagramas de estados Subestados



Sistema de control de temperatura

## Diagramas de estados Subestados

- Una máquina de estados describe los aspectos dinámicos de un objeto cuyo comportamiento actual depende de su pasado
- A menos que se especifique lo contrario, cuando una transición entra en un estado compuesto, la acción de la máquina de estados anidados comienza de nuevo en su estado inicial, a menos que la transición de dirija a un subestado

## Diagramas de estados Subestados

- Hay veces que deseamos que un objeto recuerde el último subestado antes de abandonar el estado
- Con este fin se puede tener subestados con *historia*
  - *Historia superficial*, un nivel de anidamiento. (H)
  - *Historia profunda*, cualquier nivel de anidamiento. (H)\*

## Diagramas de estados Subestados

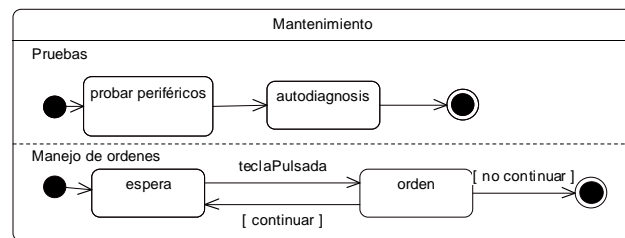
- Los *subestados secuenciales* son el tipo de máquinas de estados anidados que aparecen más frecuentemente cuando se modela
- Sin embargo, en ciertas situaciones de modelado es preciso especificar subestados *concurrentes*

## Diagramas de estados Subestados

- Los *subestados concurrentes* permiten especificar dos o más máquinas de estados que se ejecutan en paralelo en el contexto del objeto que las contiene
- Si un subestado concurrente alcanza su estado final antes que el otro, el control en ese subestado espera en su estado final

## Diagramas de estados Subestados

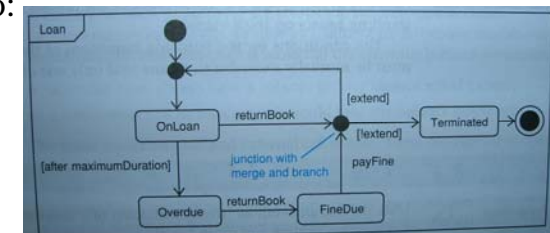
- Ejemplo:



Subestados concurrentes

## Diagramas de estados Otros elementos

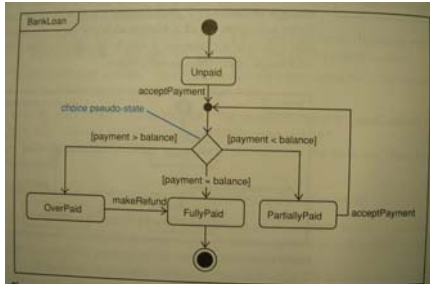
- Pseudo-estado de unión
  - Representa puntos donde las transiciones se unen o bifurcan
  - Ejemplo:



Pseudo-estado de unión

## Diagramas de estados Otros elementos

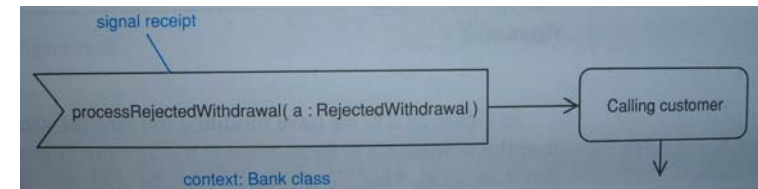
- Pseudo-estado de elección
  - Permite mostrar una bifurcación sin mezcla
  - Ejemplo:



Pseudo-estado de elección

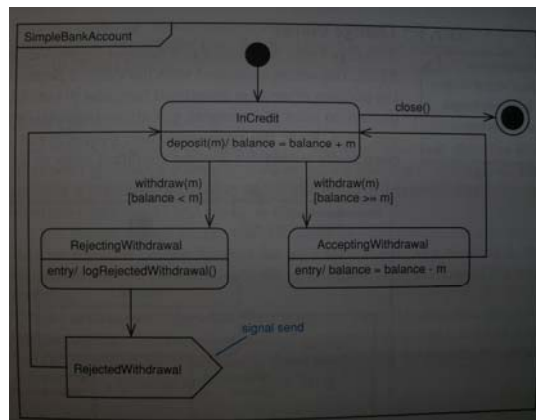
## Diagramas de estados Otros elementos

- Eventos de señal
  - Permiten caracterizar el envío/recepción de señales
  - Ejemplo:



Estado de recepción de señal

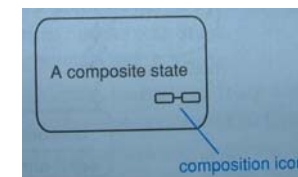
## Diagramas de estados Otros elementos



Estado de envío de señal

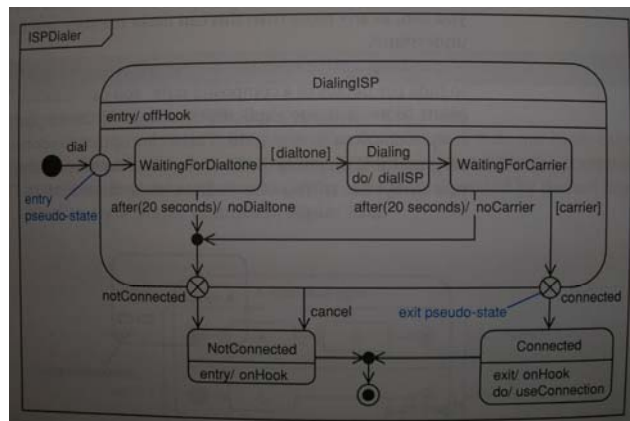
## Diagramas de estados Otros elementos

- Los *estados compuestos* permiten especificar un estado con estados anidados
  - Pueden ser:
    - Simples: una única región
    - Compuestos: dos o más regiones (conurrencia)
  - Ejemplo:



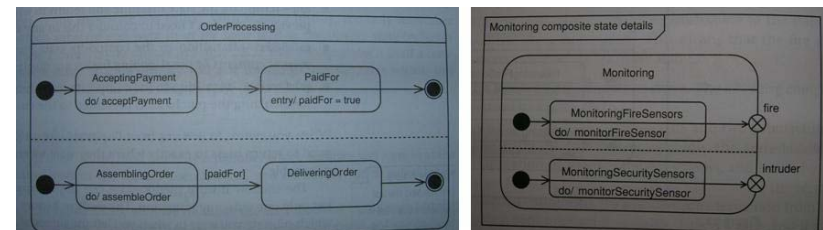
Estado compuesto

## Diagramas de estados Otros elementos



Estado compuesto simple

## Diagramas de estados Otros elementos

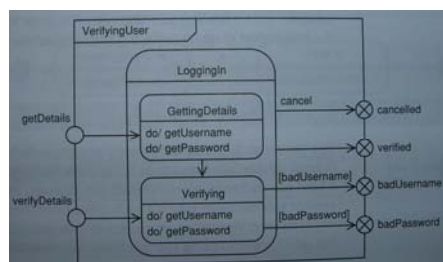


Estado compuesto ortogonal con y sin sincronización final

## Diagramas de estados Otros elementos

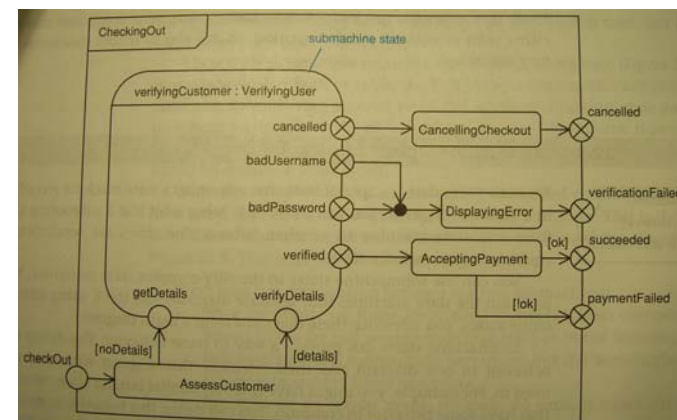
- Un *estado de submáquina* es un estado que referencia a una máquina de estados de otro diagrama

– Ejemplo:



Máquina de estados referenciada

## Diagramas de estados Otros elementos



Referencia a máquina de estados



## Diagramas de estados

### Tipos de máquina

- UML distingue entre máquinas de estados:
  - De comportamiento: para describir los detalles del ciclo de vida de una clase
  - De protocolo: para definir protocolos de operaciones (condiciones de invocación, resultados de operación, ordenación de operaciones)

## Diagramas de estados

### Tipos de máquina

- Ambos diagramas utilizan la misma notación
- Normalmente se utiliza máquinas de estado de comportamiento para definir el comportamiento de las clases

## Diagramas de estados

### Maq. est. de protocolo

- Las *máquinas de estados de protocolo* se centran únicamente en las transiciones de estados y las reglas que gobiernan el orden de ejecución de las operaciones
- Pueden utilizarse para describir las implementaciones de interfaces o puertos
- No definen el comportamiento dentro de estados

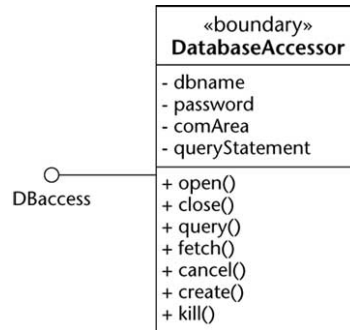
## Diagramas de estados

### Maq. est. de protocolo

- Una máquina de protocolo bien formada incluye todas las transiciones legales para cada operación.
- Las transiciones tienen la sintaxis:  
`[precondición] lista eventos / [postcondición]`

# Diagramas de estados Maq. est. de protocolo

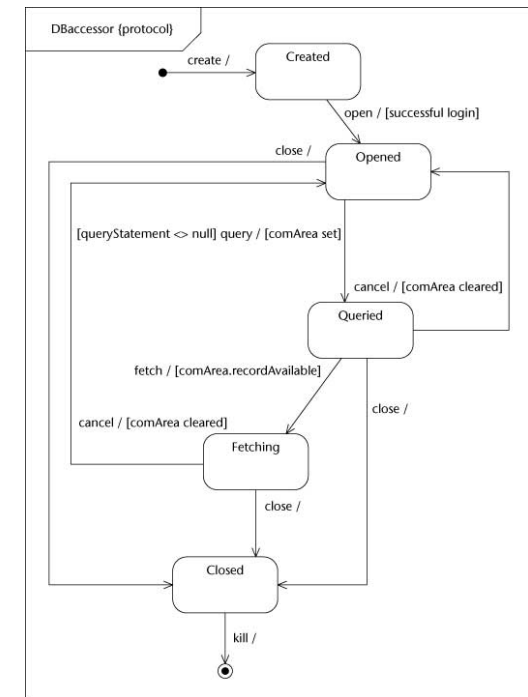
- Ejemplo:



Implementación del interfaz *DBAccess* por *DatabaseAccesor*

# Diag... Maq...

Protocolo de  
uso del  
interfaz  
*DBAccess*



## Paquetes Definición

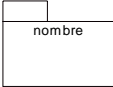

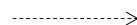
- Un *paquete* es un mecanismo de propósito general para organizar elementos en grupos
- Los paquetes se pueden anidar dentro de otros paquetes
- Un sistema puede corresponder a un único paquete de alto nivel, con todo el resto del modelo contenido recursivamente en él

## Paquetes Definición

- En un paquete pueden aparecer tanto elementos del modelo como diagramas UML

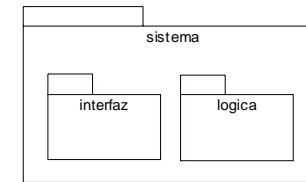
## Paquetes Notación

- Los elementos más comunes relacionados con los paquetes son:

- Paquete: 
- Generalización: 
- Dependencia: 

## Paquetes Notación

- Ejemplo:



Paquetes anidados

## Paquetes Características

- Un paquete puede contener:
  - Clases.
  - Interfaces.
  - Componentes.
  - Nodos.
  - Colaboraciones.
  - Casos de uso.
  - Diagramas.
  - Otros paquetes.

## Paquetes Características

- La posesión es una *relación compuesta*:
  - El elemento se declara en el paquete.
  - Si el paquete se destruye, también se destruye el elemento.
  - Cada elemento pertenece exclusivamente a un único paquete.

## Paquetes Características

- Un paquete determina un *espacio de nombres*, lo que quiere decir que los elementos de la misma categoría deben tener nombres únicos en el contexto del paquete contenedor
  - De esta forma, `P1::Venta` y `P1::Venta` es erróneo, pero no lo es, `P1::Venta` y `P2::Venta`.

## Paquetes Características

- Elementos de distinto tipo (e.g. clases y componentes) sí pueden tener el mismo nombre dentro de un paquete.
- Los paquetes pueden tener *paquetes anidados*
  - Esto conlleva añadir los nombres a los elementos, e.g. `P1::P2::Venta`.

## Paquetes Características

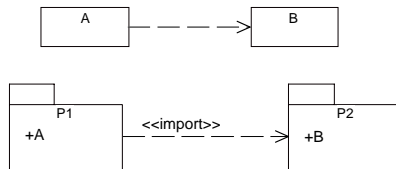
- Es mejor limitar el anidamiento a dos o tres niveles y utilizar la importación para organizar paquetes.
- Además, los paquetes permiten especificar *visibilidad* para sus elementos:
  - *Público* (+): accesible dentro y fuera.
  - *Protegido* (#):: accesible por los paquetes hijo.
  - *Privado* (-): no accesible fuera del paquete.

## Paquetes Relaciones

- Las relaciones que se pueden establecer entre paquetes son:
  - Importación.
  - Acceso.
  - Generalización

## Paquetes Relaciones

- La *importación* tiene sentido en situaciones:



Importación entre paquetes

## Paquetes Relaciones

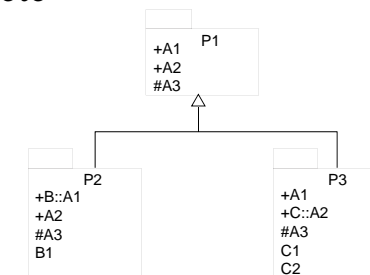
- Las partes públicas de un paquete son sus *exportaciones* (e.g. la clase B).
- Las partes que exporta un paquete sólo son visibles al contenido de aquellos paquetes que lo importan explícitamente.
- Las dependencias de importación y acceso no son transitivas.

## Paquetes Relaciones

- El *acceso* es similar a la importación, pero a diferencia de ésta, no añade el contenido del paquete destino al espacio de nombres, y por lo tanto, hay que calificar los elementos con el nombre del paquete accedido
  - Se denota con el estereotipo <<access>>.
  - En la práctica, es más común la importación.

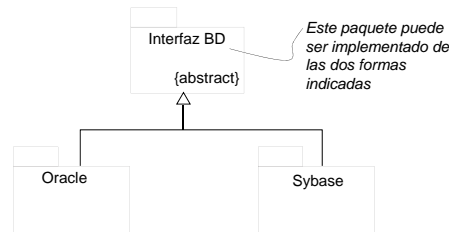
## Paquetes Relaciones

- La *generalización de paquetes* representa la generalización de los elementos contenidos en el paquete
- Ejemplos:



Generalización entre paquetes

## Paquetes Relaciones



Generalización entre paquetes

## Paquetes Relaciones

- Los paquetes implicados en las generalizaciones siguen el mismo principio de sustitución que las clases.
- Así, un paquete especializado puede usarse dondequiera que se use un paquete más general.

## Paquetes Estereotipos

- UML define cinco estereotipos estándar que se aplican a los paquetes:
  - *facade*. Especifica un paquete que es sólo una vista de algún otro paquete.
  - *framework*. Especifica un paquete que consta principalmente de patrones.
  - *stub*. Especifica un paquete que sirve de *proxy* para el contenido público de otro paquete.

## Paquetes Estereotipos

- *subsystem*. Especifica un paquete que representa una parte independiente del sistema completo que se está modelando.
- *system*. Especifica un paquete que representa el sistema completo que se está modelando.

## Diagramas de componentes

### Definición

- Un *diagrama de componentes* muestra las partes internas, los conectores y los puertos que implementan un componente
- Un *componente* representa una parte modular de un sistema que encapsula sus contenidos y cuya manifestación es reemplazable dentro de su entorno

## Diagramas de componentes

### Definición

- Un componente actúa como una caja negra cuyo comportamiento externo está completamente definido por sus interfaces proporcionados e implementados
- A causa de esto, un componente puede ser reemplazado por cualquier otro que soporte su mismo protocolo

## Diagramas de componentes

### Definición

- Los componentes pueden representar:
  - Algo que puede ser instanciado en ejecución, como un EJB.
  - Una construcción lógica como un subsistema
- Los componentes:
  - Se manifiestan físicamente como uno o más *artefactos*
  - Pueden tener atributos y operaciones
  - Pueden participar en relaciones de asociación y generalización

## Diagramas de componentes

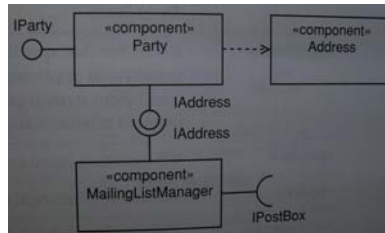
### Definición

- Aunque la notación no varía en exceso con respecto a UML 1.x, su semántica sí:
  - Componentes UML 1.x están más cercanos al concepto de *artefacto*
  - Componentes UML 2.x elementos del diseño caracterizados por sus puertos, y que en ejecución se instanciarán (e.g. EJBs) o se corresponderán con varios objetos (e.g. subsistemas).

## Diagramas de componentes

### Definición

- Ejemplo:



Componentes UML 2.x

## Diagramas de componentes

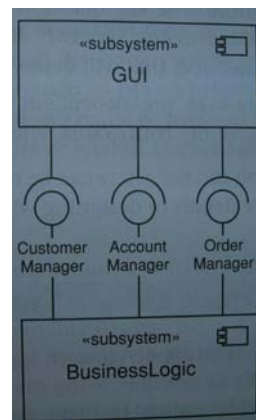
### Subsistemas

- Un *subsistema* es un componente que actúa como una unidad de descomposición de un sistema mayor
- Los subsistemas no pueden ser instanciados en ejecución, pero sí sus contenidos
- Los subsistemas son claves para el diseño de una aplicación
- Los interfaces conectan subsistemas para crear la arquitectura del sistema

## Diagramas de componentes

### Subsistemas

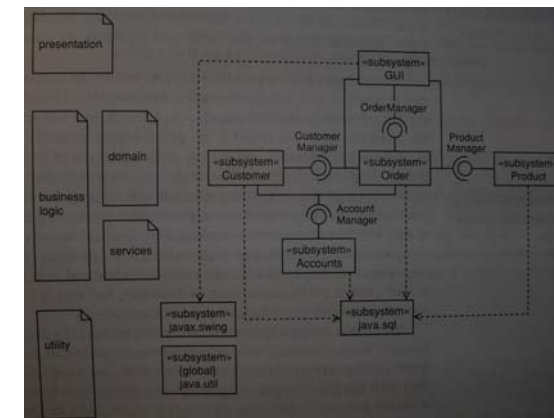
- Ejemplo:



Subsistemas

## Diagramas de componentes

### Subsistemas



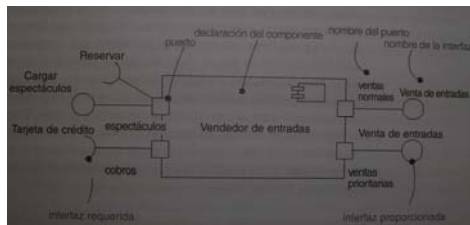
Subsistemas



## Diagramas de componentes

### Puertos

- Los puertos suelen utilizarse en diagramas de componentes
- La semántica es idéntica a la de clases
- Ejemplo:



Puertos en un componente

## Colaboraciones

- Una *colaboración* es una sociedad de clases, interfaces y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de los elementos

## Colaboraciones

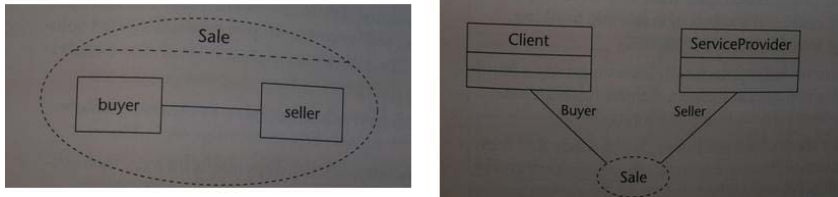
- Una colaboración muestra una solución dependiente de una implementación interna de un caso de uso en términos de las relaciones entre clases y objetos (el *contexto* de la colaboración) y su interacción para conseguir la funcionalidad deseada (la *interacción* de la colaboración).

## Colaboraciones

- Una colaboración se denota por una elipse discontinua
- Para explicar una colaboración son necesarios distintos diagramas que reflejen tanto el contexto como la interacción entre los elementos de la colaboración

## Colaboraciones

- Ejemplos:


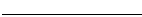
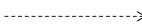


Colaboraciones

## Diagramas de despliegue Definición

- Un *diagrama de despliegue* es un diagrama que muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos
- Un *nodo* es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional que generalmente tiene alguna memoria, y a menudo, capacidad de procesamiento

## Diagramas de despliegue Notación

- Los elementos más comunes de los diagramas de despliegue son:
  - Nodo: 
  - Asociación: 
  - Dependencia: 

## Diagramas de despliegue Notación

- La asociación denota conexión física entre nodos
- La dependencia se establece entre los elementos que pudieran aparecer dentro de los nodos
- Ejemplos:

## Diagramas de despliegue Notación

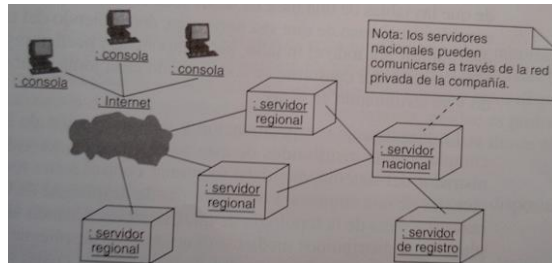


Diagrama de despliegue para un sistema distribuido

## Diagramas de despliegue Notación

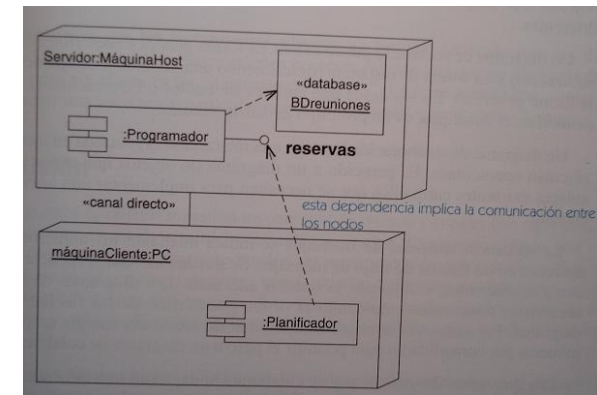


Diagrama de despliegue con componentes

## Diagramas de despliegue Notación

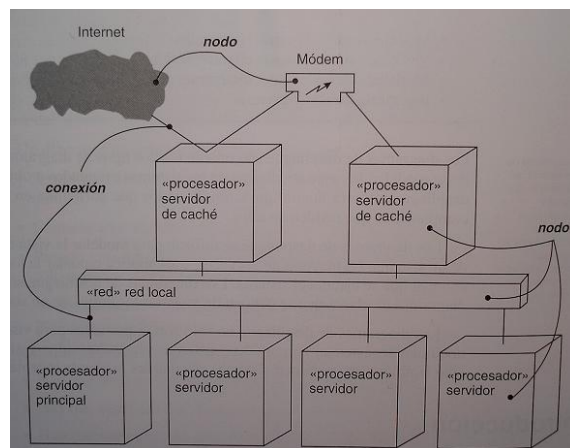


Diagrama de despliegue

## Diagramas de despliegue Notación

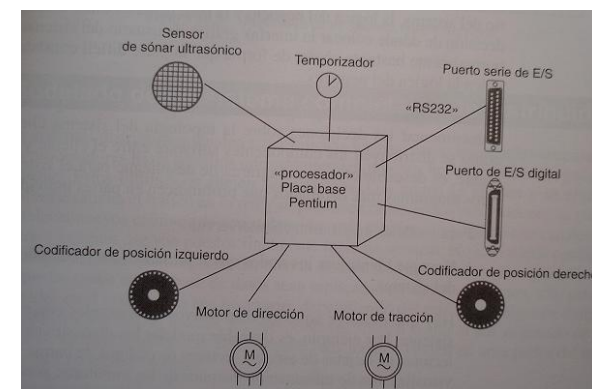


Diagrama de despliegue para un sistema empujado

## Diagramas de despliegue Notación

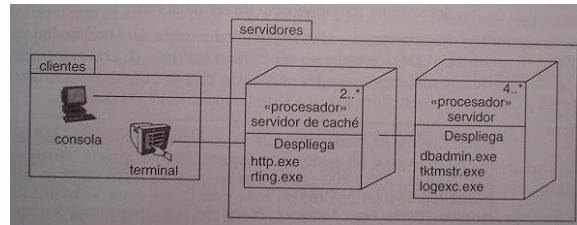


Diagrama de despliegue con paquetes

## Diagramas de despliegue Notación

- También incluyen:
  - Dispositivos
  - Entornos de ejecución
  - Instancias de nodos
  - Artefactos
  - Instancias de artefactos

## Diagramas de despliegue Notación

- Un *dispositivo* es un nodo estereotipado (device) que representa un dispositivo físico (e.g. un ordenador PC)
- Un *entorno de ejecución* es un nodo estereotipado (execution environment) que representa un entorno de ejecución de software (e.g. Apache web server)

## Diagramas de despliegue Notación

- Ejemplo:

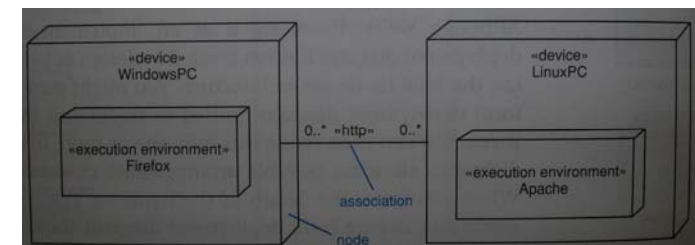
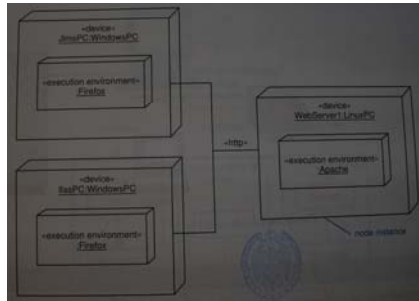


Diagrama de despliegue con entornos de ejecución

## Diagramas de despliegue

### Notación

- Una *instancia de nodo* caracteriza una instancia específica de un nodo
- Ejemplo:



Instancias de nodos

## Diagramas de despliegue

### Artefactos

- Un *artefacto* representa una especificación de un elemento concreto del mundo real
- Ejemplos de artefactos son:
  - Ficheros fuente.
  - Ficheros ejecutables.
  - Scripts.
  - Tablas de bases de datos.
  - Documentos.

## Diagramas de despliegue

### Artefactos

- Por su naturaleza, los artefactos pueden aparecer tanto en diagramas de componentes, como en diagramas de despliegue
- Una *instancia de artefacto* es una instancia particular de un artefacto concreto en una determinada instancia de nodo

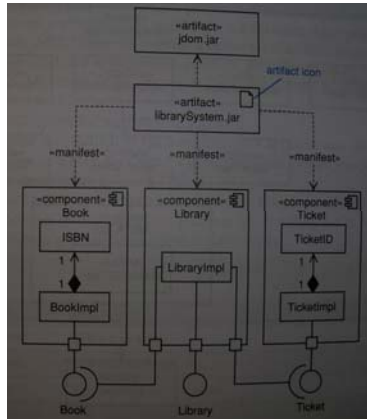
## Diagramas de despliegue

### Artefactos

- Un artefacto puede proporcionar la manifestación física de cualquier elemento UML
- Dicha manifestación se representa como una dependencia estereotipada con *manifest*
- Los artefactos pueden presentar dependencias entre ellos

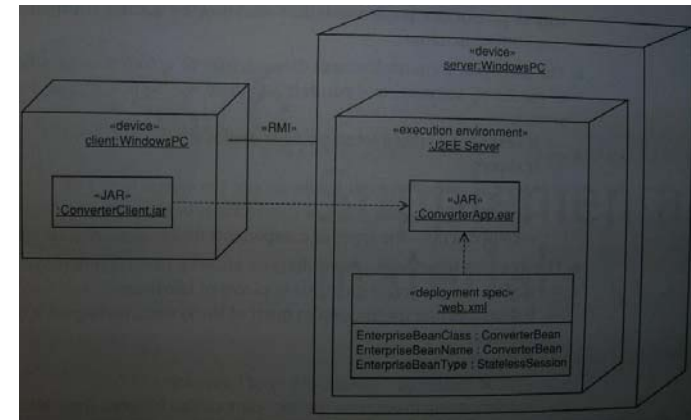
## Diagramas de despliegue Artefactos

- Ejemplos:



Artefacto

## Diagramas de despliegue Artefactos



## Diagramas de despliegue Artefactos

- UML 2.x define los siguientes artefactos:
  - file: archivo físico
  - deployment spec: especificación de detalles de despliegue (e.g. web.xml en J2EE)
  - document: fichero genérico que contiene información
  - executable: fichero ejecutable
  - library: librería estática o dinámica
  - script: script ejecutable por un intérprete
  - source: fichero compilable en ejecutable

## Diagramas de estr. compuesta Definición

- Un *diagrama de estructura compuesta* muestra la estructura interna de una clase o colaboración
- Muestra relaciones en ejecución
- Suelen usarse en diagramas de clases o componentes
- Su notación se ajusta a la de estos diagramas

## Diagramas de estr. compuesta

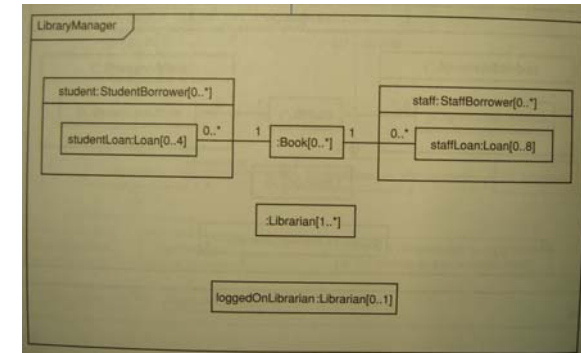
### Clases

- Un diagrama de estructura compuesta de clases hace explícitos los papeles que las instancias juegan en el contexto de una clase

## Diagramas de estr. compuesta

### Clases

- Ejemplo:



Estructura compuesta de clase

## Diagramas de estr. compuesta

### Componentes

- En el contexto de diagramas de componentes, estos diagramas pueden utilizarse para hacer explícita la estructura interna del componente
- Dentro de un componente puede haber otros componentes relacionados a través de *conectores* entre puertos

## Diagramas de estr. compuesta

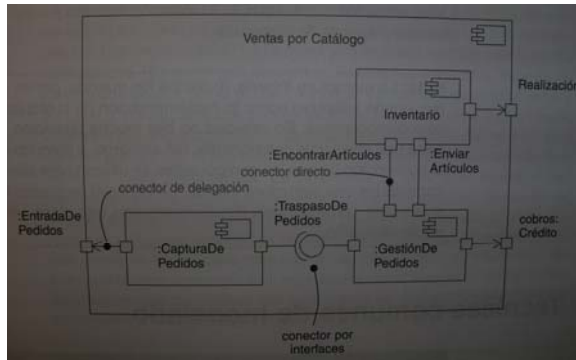
### Componentes

- Estos conectores pueden ser:
  - *Directos*: los componentes se enlazan de manera explícita, directamente o a través de puertos
  - *Por interfaces*: los componentes se enlazan a través de interfaces que realizan o necesitan
  - *Globales*: los mensajes sobre el puerto externo son delegados en el puerto interno

## Diagramas de estr. compuesta

### Componentes

- Ejemplo:



Estructura compuesta de componente

## Diagramas de tiempo

### Definición

- Un *diagrama de tiempo* muestra el cambio de estado a lo largo de una línea de vida en términos de una unidad de tiempo definida
- También puede caracterizar las restricciones temporales en interacciones entre dos o más líneas de vida

## Diagramas de tiempo

### Notación

- Ejemplo:

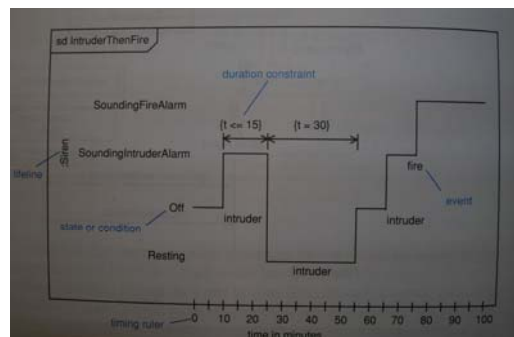


Diagrama de tiempo

## Diagramas de tiempo

### Notación

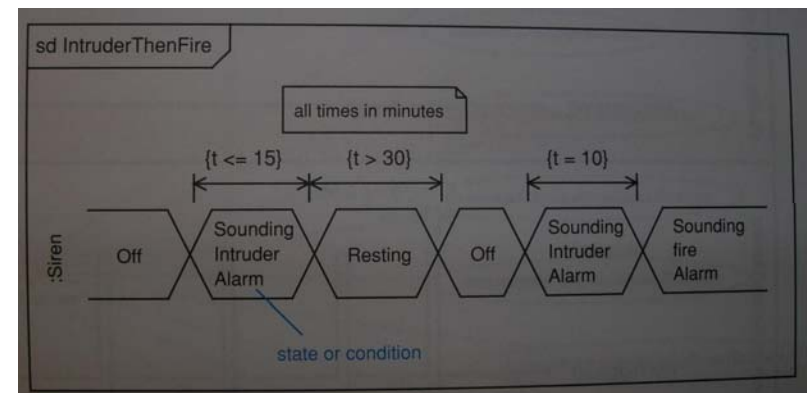
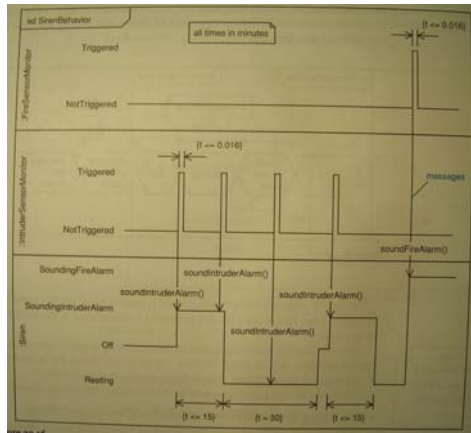


Diagrama de tiempo compacto



# Diagramas de tiempo

## Notación

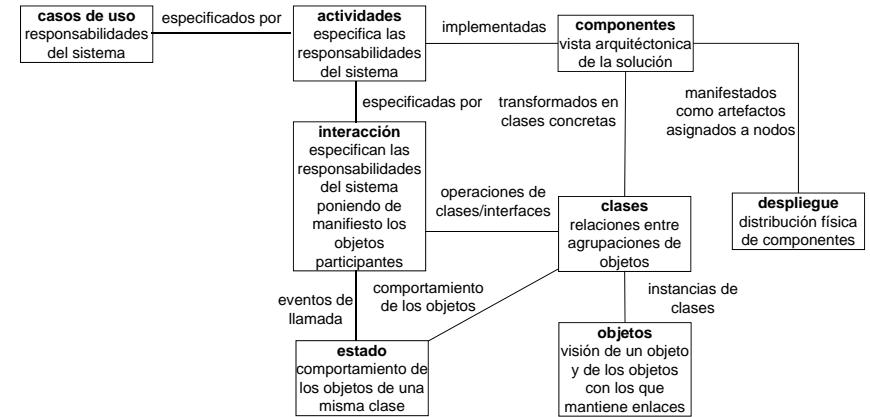


Ingeniería del Software  
Antonio Navarro

Diagrama de tiempo con interacciones

225

# Racionalidad de uso de UML

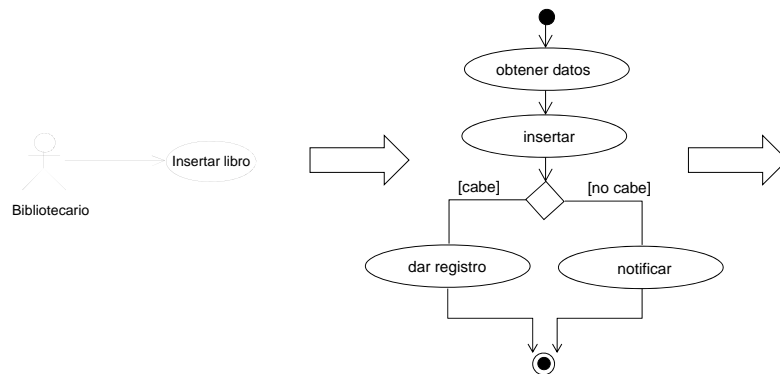


Ingeniería del Software  
Antonio Navarro

Racionalidad de uso de UML

226

# Racionalidad de uso de UML

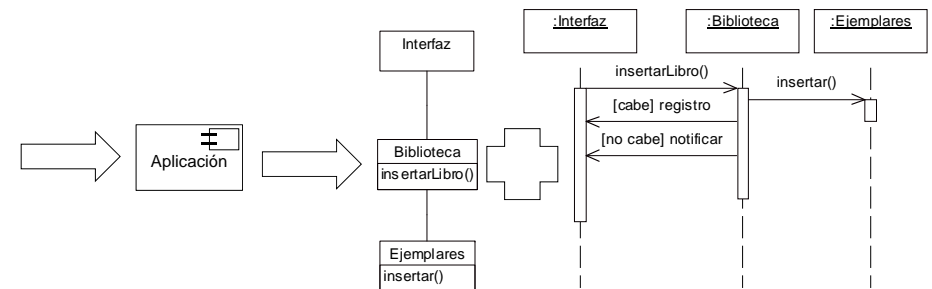


Ingeniería del Software  
Antonio Navarro

Insertar libro

227

# Racionalidad de uso de UML

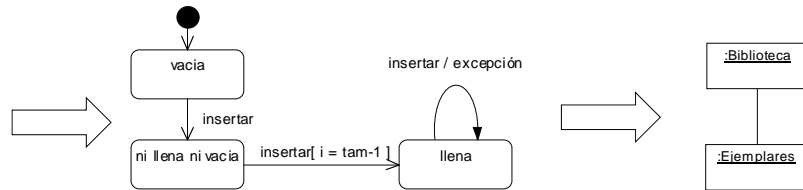


Ingeniería del Software  
Antonio Navarro

Insertar libro

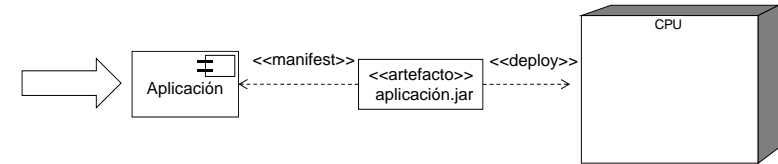
228

## Racionalidad de uso de UML



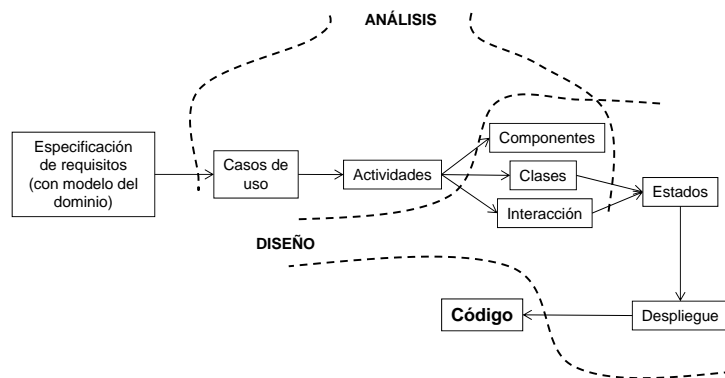
Insertar libro

## Racionalidad de uso de UML



Insertar libro

## Análisis y diseño OO



Análisis y diseño OO

## Conclusiones

- UML: notación visual
- Vista lógica vs. física
- Vista estructural vs. dinámica
- Casos de uso
- Actividades
- Clases
- Objetos

## Conclusiones

- Interacción
  - Secuencia.
  - Comunicación.
- Estados
- Componentes
- Despliegue
- Estructura compuesta
- Tiempo

## Conclusiones

- Paquetes
- Racionalidad de uso