

Unified Modeling Language



María de los Ángeles Sumano López
Juan Manuel Fernández Peña

julio 2001
Actualización 2006



Vistazo general

Naturaleza y propósitos

- ⌘ Lenguaje para el modelado visual de propósito general que es utilizado para:
 - ☒ especificar, visualizar, construir y documentar los *artefactos* de un sistema de software.
- ⌘ Captura información acerca de la estructura estática y el comportamiento dinámico del sistema.
- ⌘ Permite partir el sistema de software mediante paquetes que agrupan construcciones organizacionales.

Metas

- ⌘ Que todos los modeladores lo puedan utilizar.
- ⌘ Soportar todos los procesos de desarrollo*.
- ⌘ Ser lo más simple posible y aún así modelar el rango completo de aplicaciones prácticas.

* Jacobson, I., Booch, G., Rumbaugh, J. (1999): **The Unified Software Development Process**, Addison-Wesley
Rosemberg, D. Y Scott, K. (1999): **Use Case Driven Object Modeling with UML. A practical approach**, Addison-Wesley

Desarrollo de UML

- ⌘ Aceptación creciente de modelado de objetos
- ⌘ Existencia de muchos esfuerzos similares en contenido y forma diferente
- ⌘ UML: unifica esfuerzos de muchos
- ⌘ UML estándar del OMG
- ⌘ Ofrece notación común y tolera variantes

Conceptos (1)

- ⌘ Estructura estática.
 - ☒ Define el universo del discurso.
 - ☒ Conceptos, propiedades y relaciones.
 - ☒ Los conceptos de la aplicación son modelados como clases, su información como atributos y su comportamiento como operaciones.
 - ☒ Existen relaciones entre clases.
 - ☒ Se representa a través de diagramas de clases.

Conceptos (2)

⌘ Comportamiento dinámico.

- ☒ Modelo de interacción de un objeto con el resto del mundo a lo largo de su historia.
 - ☒ Diagramas de estados
- ☒ Modelo de patrones de comunicación de un conjunto de objetos conectados y su interacción en la implantación de un comportamiento.
 - ☒ colaboraciones

Conceptos (3)

⌘ Construcciones de implantación (despliegue) (deployment view).

- ☒ Implantación física:
 - ☒ Componente.- parte física reemplazable de un sistema.
 - ☒ Nodo.- recurso computacional que define una localización en tiempo de corrida (contiene por componentes y objetos).

Conceptos (4)

⌘ Modelo organizacional.

- ☒ División en partes coherentes (paquetes) para que diferentes equipos de personas puedan trabajar concurrentemente.
- ☒ Los paquetes pueden estar formados por partes reusables.
- ☒ La dependencia de jerarquía entre paquetes puede ser impuesta por la arquitectura del sistema.

Conceptos (5)

⌘ Mecanismos de extensión.

- ☒ Estereotipos como nuevos elementos, con:
 - ☒ restricciones (OCL)
 - ☒ valores etiquetados (variable=valor)
 - ☒ diferente interpretación
 - ☒ icono diferente

Modelos

Naturaleza y propósito

⌘ ¿Qué es un modelo?

- ☒ es la representación de algo (su abstracción).
- ☒ Sólo captura lo importante desde un cierto punto de vista omitiendo lo que no es relevante en éste.
- ☒ Se expresa en un medio que convenga trabajar.

⌘ Un modelo de un sistema de software se hace con un lenguaje de modelado. Está formado por semántica y notación (gráfica y textual).

¿Para qué hacer modelos? (1)

- ⌘ Para capturar y precisar los requerimientos y el dominio del conocimiento con el fin de llegar a acuerdos entre los involucrados.
- ⌘ Para pensar en el diseño del sistema.
- ⌘ Para registrar decisiones del diseño en una forma modificable, separada de los requerimientos.

¿Para qué hacer modelos? (2)

- ⌘ Para generar productos utilizables (declaración de clases, interfaces de usuario, BD, ...).
- ⌘ Para organizar, encontrar, filtrar, recuperar, examinar y editar la información de sistemas grandes.
- ⌘ Para explorar económicamente múltiples soluciones.
- ⌘ Para ir comprendiendo un sistema complejo.

Niveles de modelado

⌘ Detalle cambia con propósito; seguir uno:

- ☒ Guía para pensar (dejar solo el último)
- ☒ Especificación abstracta
- ☒ Especificación final, completa
- ☒ Ejemplos de sistemas particulares o partes


Contenido del modelo

- ⌘ Semántica: elementos con significado, unidos según sintaxis; para validar o generar código
- ⌘ Presentación: aspectos visuales para humanos
- ⌘ Contexto: aspectos que le dan sentido; organización y uso del modelo, suposiciones

Balance



- ⌘ Modelo es idealización
- ⌘ No existe **la** buena forma de un modelo
- ⌘ Balancear según uso y nivel:
 - ☒ abstracción vs detalle
 - ☒ especificación vs implementación
 - ☒ descripción vs instancia
 - ☒ variaciones de interpretación



Diagramas
y
vistas
de

Área Estructural

⌘ Vista Estática(Diagramas de Clases)

☒ Clase, asociación, interfaz, generalización, dependencia, realización.

⌘ Vista Casos de Uso(Diagramas de Casos de Uso)

☒ Caso de uso, actor, asociación, extensión, inclusión.

⌘ Vista de Implantación(Diagramas de Componentes)

☒ Componente, interfaz, dependencia, realización.

⌘ Vista de Despliegue(Diagramas de Despliegue)

☒ Nodo, componente, dependencia, realización.

Área dinámica

⌘ Vista de Máquina de Estados (Diag. de estados)

☒ Estado, evento, transición, acción.

⌘ Vista de Actividades (Diagrama de Actividades)

☒ Estado, actividad, transición terminal, bifurcación, reunión.

⌘ Vista de Interacción (Diagramas de Secuencia y colaboración)

☒ Interacción: objeto, mensaje, activación.

☒ Colaboración: interacción, papel de colaboración, mensaje.

Área de administración del modelo

- ⌘ Vista de Administración del modelo (Diagrama de Clases)
 - ☒ Paquete, subsistema, modelo.

Área de Extensibilidad

- ☒ Todas las vistas y diagramas
 - ☒ Restricciones, estereotipos, valores etiquetados.



Vista de casos de uso

Vista de casos de uso

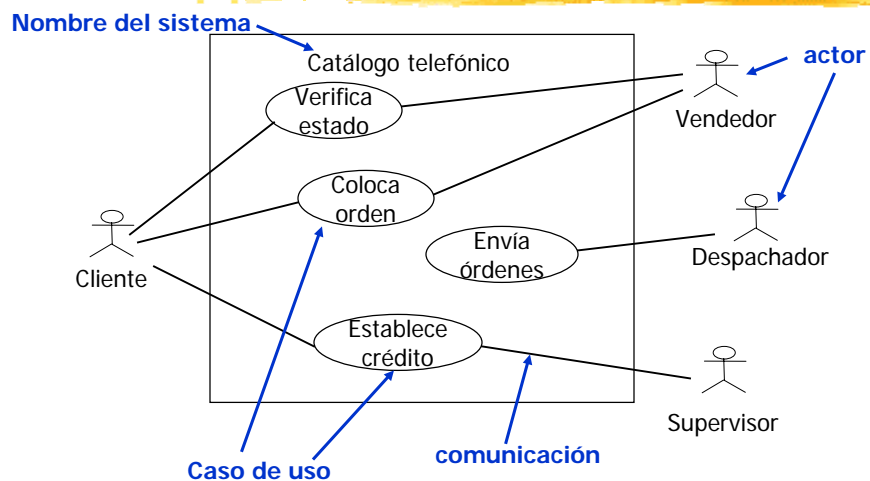


- ⌘ Captura comportamiento de sistema, subsistema o clase, vista desde fuera
- ⌘ Particiona funcionalidad en transacciones que tienen sentido para actores (casos de uso)
- ⌘ Se describe con diagramas de secuencia que involucran a los actores

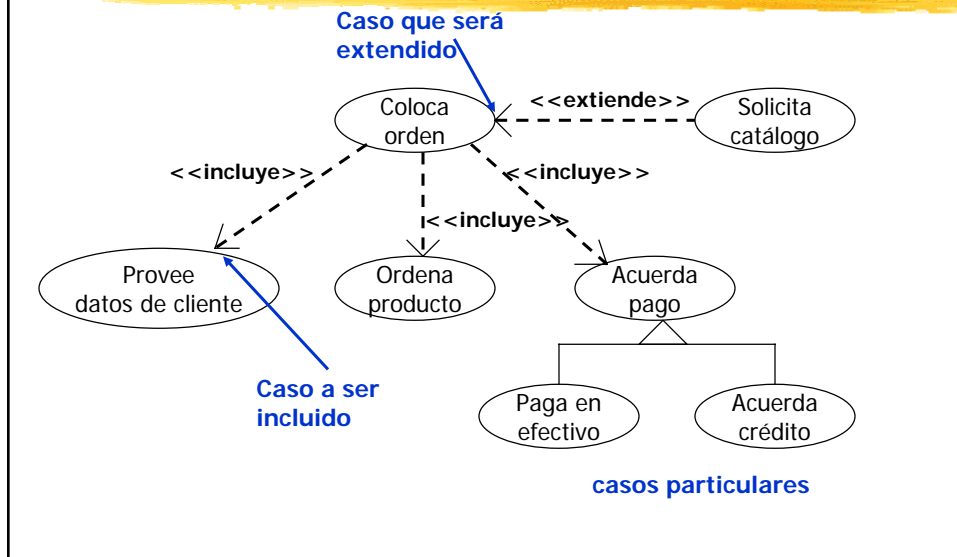
Casos de uso

- ⌘ Actor: usuario idealizado
- ⌘ Caso de uso: Unidad coherente de funcionalidad visible externamente
- ⌘ Relaciones:
 - ☒ asociación
 - ☒ extensión
 - ☒ inclusión
 - ☒ generalización

Casos de uso



Casos de uso



Casos de uso: descripción textual

Caso de uso

Actor acción

Respuesta del sistema

Actor acción

Respuesta del sistema

...

Camino alternativo

...

Casos de uso: descripción textual

Caso de uso: **consulta**

Investigador oprime "Consultar"

Se muestra lista de temas

Investigador selecciona una y da enter

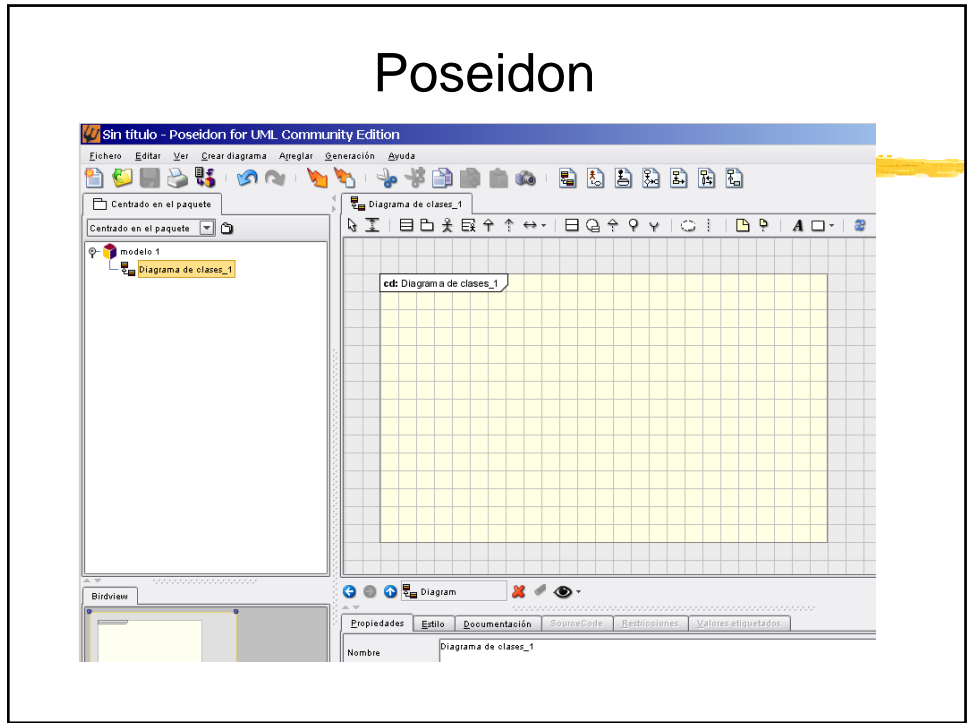
Se muestra una ventana con información del tema

Preparación de diagramas

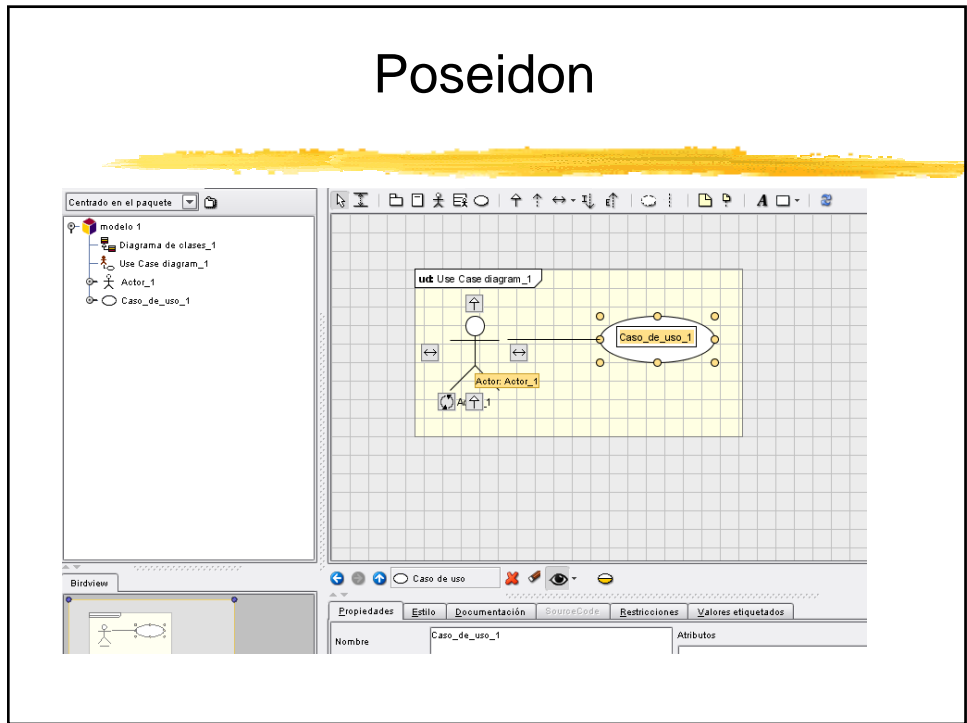
Herramientas disponibles:

- Poseidon (Gentleware) Community Edition
- Select Component Factory
- Rational Rose

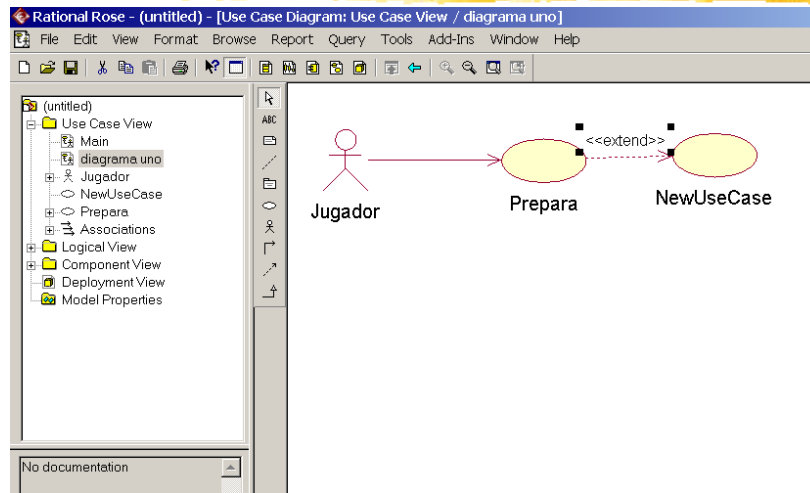
Poseidon



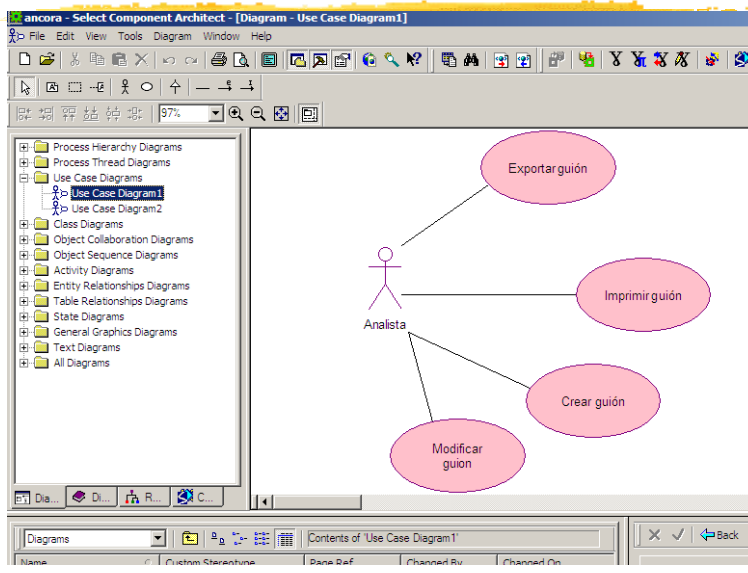
Poseidon



Rational Rose



Select Component Factory



Vista Estática

Vista Estática

- ⌘ Fundamento de UML.
- ⌘ Elementos de la vista estática: conceptos que son significativos en una aplicación, incluyendo conceptos del mundo real, abstractos, de implementación y computacionales.
- ⌘ Los elementos importantes en la vista estática son *clasificadores* y sus *relaciones*.








Vista Estática

- ⌘ Un diagrama de clases muestra el conjunto de clases y objetos importantes que hacen parte de un sistema, junto con las relaciones existentes entre estas clases y objetos.
- ⌘ Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con las demás en el modelo.

Clasificadores

- ⌘ Un clasificador es un elemento modelado que describe una cosa.
- ⌘ Hay varios tipos de clasificadores, incluso de clases, interfaces y tipos de datos.
- ⌘ Un clasificador es un concepto discreto en el modelo y tienen identidad, estado, conducta y relaciones.

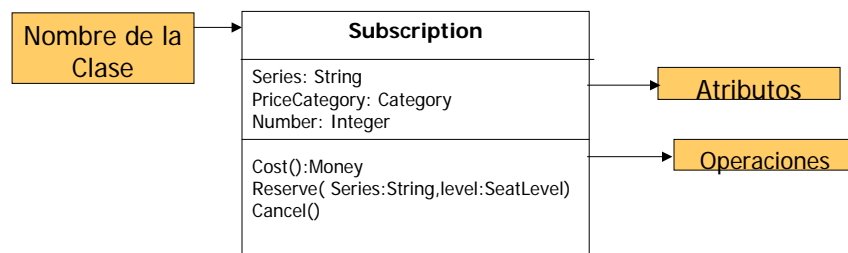
Algunos Clasificadores

Clasificador	Función	Notación
Actor	Persona o elemento externo al sistema	
Clase	Concepto del sistema modelado	
Caso de uso	Especifica comportamiento hacia exterior	
Componente	Parte física del sistema	
Interfaz	Conjunto de operaciones con nombre	
Nodo	Recurso computacional	
Paquete	Agrupar elementos para su manejo	

Clase

⌘ Representada por un rectángulo con tres divisiones internas, son los elementos fundamentales del diagrama.

⌘ Una clase describe un conjunto de objetos con características y comportamiento idéntico.



Objeto

⌘ Notación similar a la de clase

⌘ nombre:

⌘ clase (objeto genérico)

⌘ papel:clase (objeto definido por su papel)

⌘ nombre:clase (objeto propio)

Atributo

⌘ Identifican las características propias de cada clase.

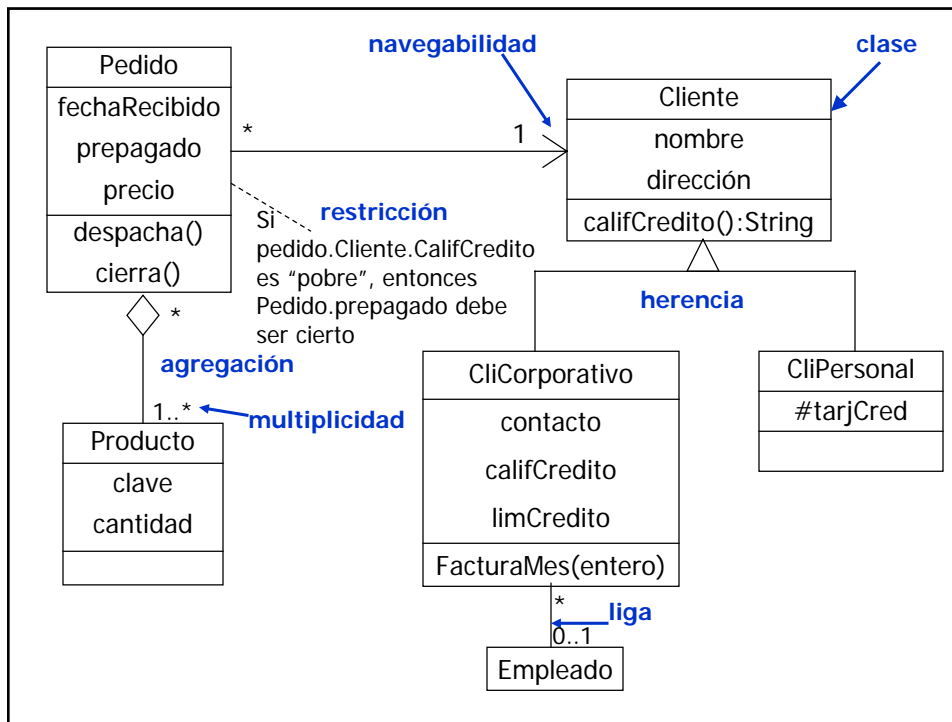
⌘ Generalmente son de tipos simples, ya que los atributos de tipos compuestos se representan mediante asociaciones de composición con otras clases.

Operación

- ⌘ El conjunto de operaciones describen el comportamiento de los objetos de una clase.

Relaciones

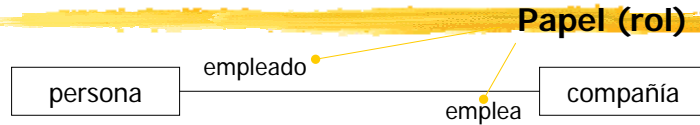
- Relaciones entre los clasificadores:
 - asociación
 - agregación, composición, liga (enlace)
 - generalización
 - herencia, realización
 - dependencia
 - acceso, importación, refinamiento, trazado, etc.



Asociaciones

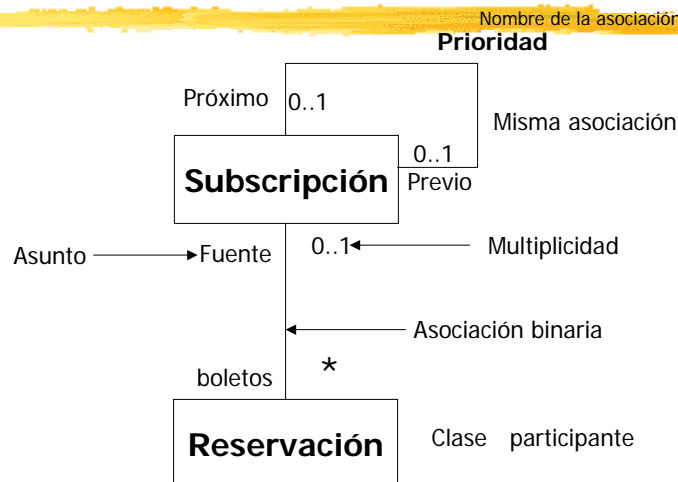
- Una asociación describe conexiones entre objetos u otras instancias de un sistema.
- Se refiere a una lista ordenada de dos o más clasificadores con repeticiones permitidas.
- Las más común es la asociación binaria entre un par de clasificadores.
- Su propiedad más importante es la multiplicidad: cuantas instancias de una clase pueden ser asociadas a una instancia de otra clase.

Asociaciones

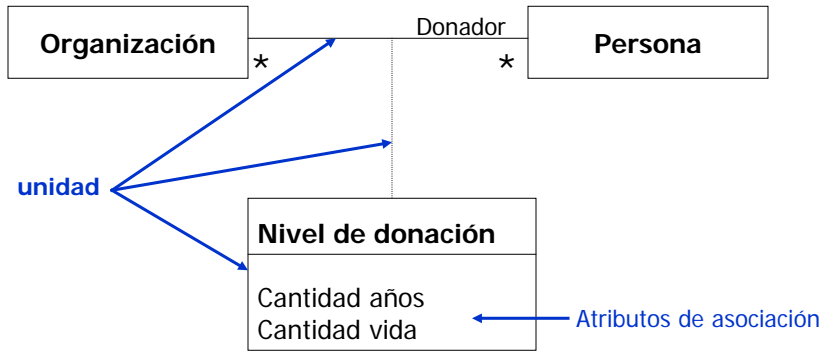


- Una asociación puede tener atributos propios
- Se pueden tener asociaciones entre objetos de la misma clase.
- Una asociación que conecta a dos clases es llamada asociación binaria.
- Asociaciones que conectan a mas de dos clases son llamadas n-arias.
- Se usan las asociaciones cuando se quiere mostrar relaciones estructurales.

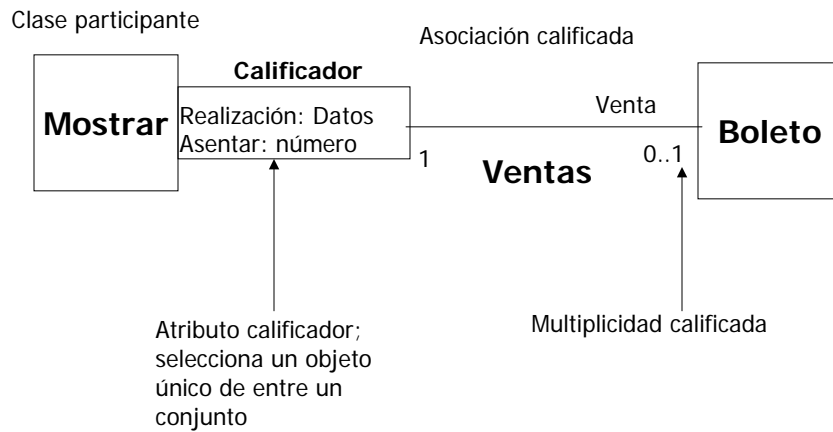
Notación de asociación



Clase de asociación



Asociación calificada

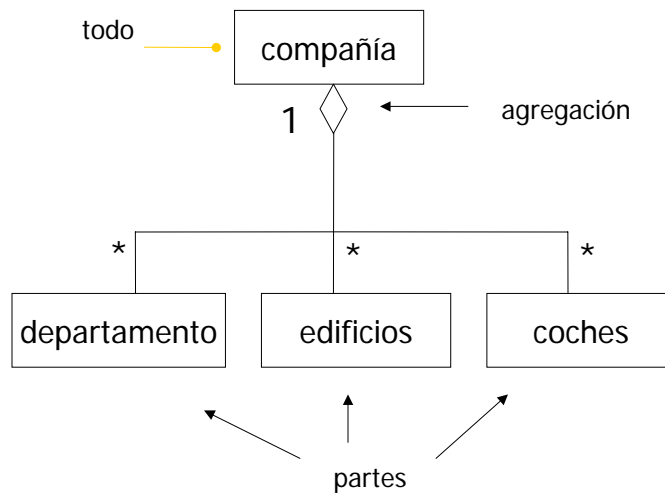


Agregación

⌘ Cuando se quiere modelar una relación "todo/parte", en la cual una clase representa algo más grande (el "todo"), y consiste de cosas más pequeñas (las "partes").

⌘ Esta representa una relación de tipo "tiene-un", lo cual significa que un objeto del "todo" tiene objetos de la "parte"

Agregación



Composición

- ⌘ Es una asociación más fuerte en la cual el compuesto tiene absoluta responsabilidad de dirigir sus partes como distribución y redistribución.
- ⌘ Se caracteriza por estar conectada por un diamante relleno. ◆
- ⌘ No dar mucha importancia.

Relación de generalización

- Es de tipo taxonómico, es decir, para clasificar
- Dos usos básicos:
 - Puede representar principio de sustitución (B. Liskov) entre tipos (variables); polimorfismo
 - Descripción incremental (herencia)

Generalización

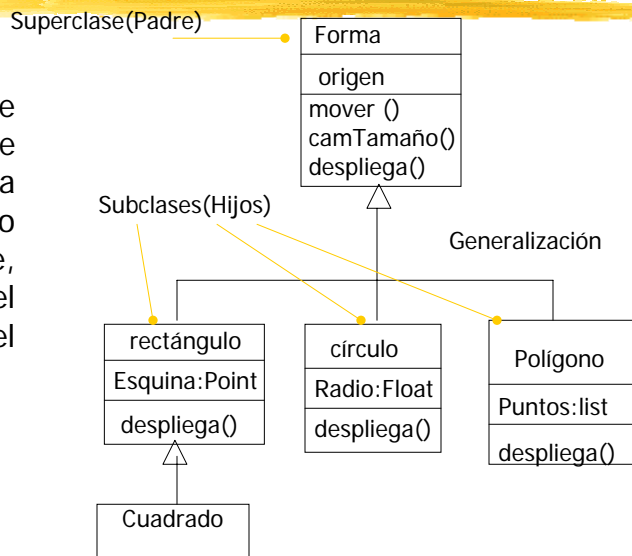
- ⌘ Es una relación entre una cosa general (llamada superclase o padre) y una clase más específica de esa cosa (llamada subclase o hijo).
- ⌘ Los objetos de subclase pueden ser usados dondequiera que la superclase pueda aparecer, pero no al revés.
- ⌘ Superclase puede ser abstracta (pospone implementación)
- ⌘ Se aplica también a los demás clasificadores

Herencia

- ⌘ Un hijo hereda las propiedades del padre, especialmente sus atributos y servicios.
- ⌘ Generalmente, pero no siempre, el hijo tiene atributos y servicios además de los que se encuentran en sus padres.
- ⌘ Una clase puede tener uno o más padres.
- ⌘ Una clase que tiene un padre se dice que usa herencia simple.

Generalización

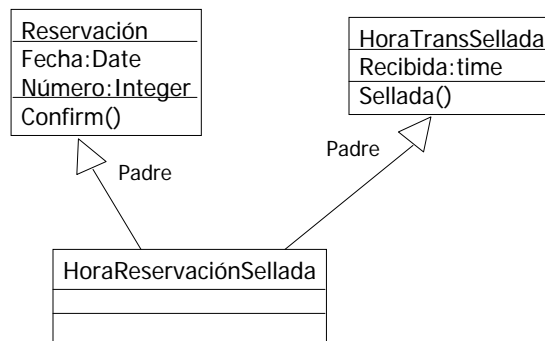
⌘ Un servicio de un hijo que tiene la misma firma que un servicio del padre, invalida el servicio del padre.



Herencia múltiple

⌘ Una clase con más de un padre se dice que usa herencia múltiple.

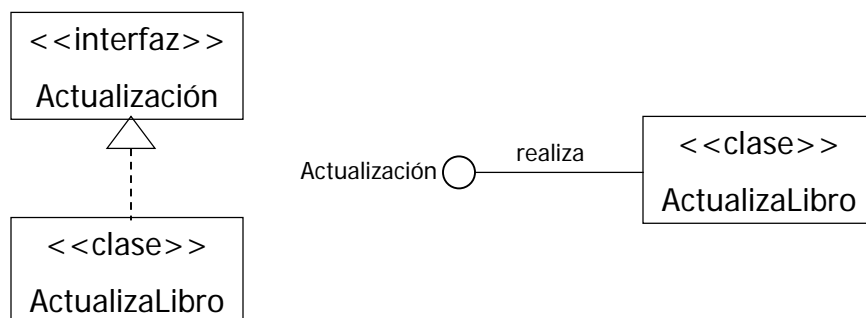
⌘ El hijo hereda los atributos y servicios de cada padre. Riesgo de conflicto.



Relación de realización

- Relaciona una especificación con una aplicación. Una interfaz es una especificación de conducta sin la aplicación. Una clase incluye estructura de aplicación.

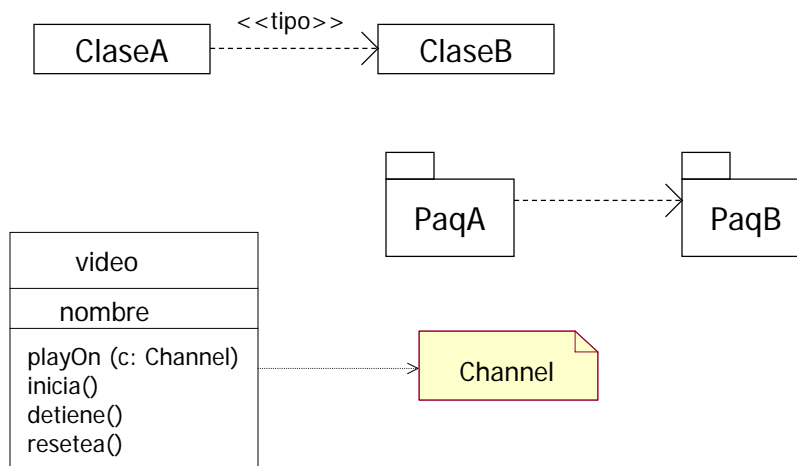
Realización



Relación de dependencia

- Una dependencia es una relación de uso que establece que un cambio en la especificación de una cosa puede afectar otra cosa pero no necesariamente lo inverso.
- Debe indicarse el tipo de dependencia, ya que hay muchos

Relación de dependencia





Vista de interacción

Comportamiento



⌘ El comportamiento se modela como interacciones entre objetos.

⌘ Dos formas:

⌘ individual: máquinas de estados

⌘ colectiva: colaboraciones

Vista de interacción

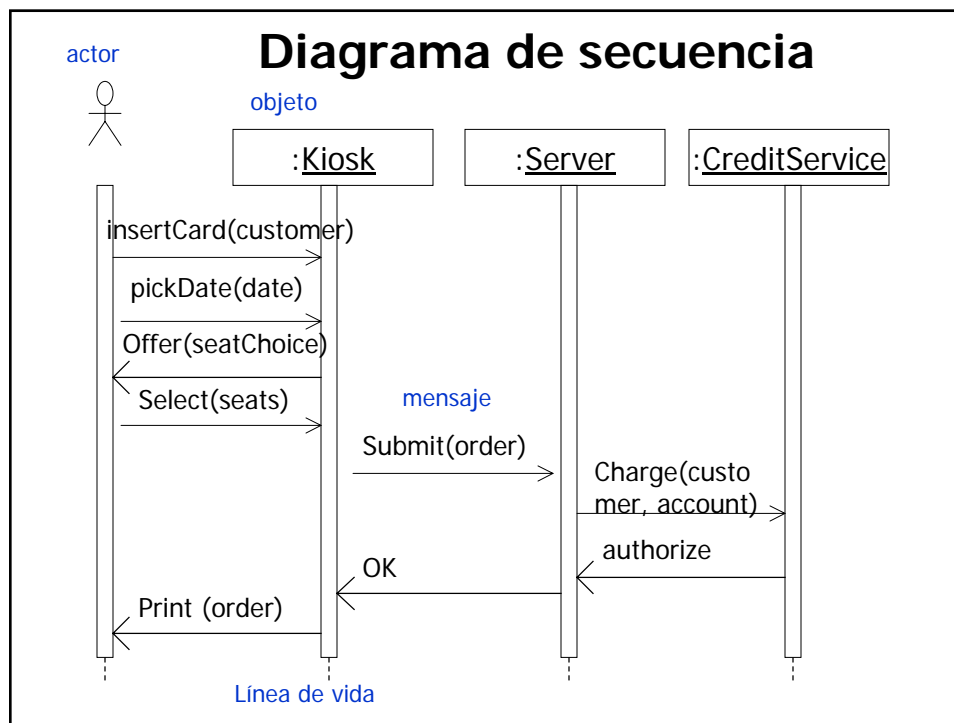
- ⌘ La vista de interacción se concreta en colaboraciones: colecciones de objetos que interactúan para implementar un comportamiento en un contexto
- ⌘ Colaboraciones en dos tipos de diagrama:
 - ☒ de secuencia
 - ☒ de colaboración

Vista de interacción

- ⌘ Las colaboraciones se descomponen en interacciones
- ⌘ Interacción: conjunto de mensajes intercambiados por clasificadores con papel a través de asociaciones con papel (es decir, debe existir conexión).
- ⌘ Mensajes: señales o llamadas; pueden tener parámetros, ser asíncronas, repetirse, ser condicionales, concurrentes y tener un orden entre ellas.
- ⌘ (El papel es importante; un objeto puede tener varios)

Diagrama de secuencia

- ⌘ Un diagrama de secuencia despliega una interacción entre un diagrama de 2 dimensiones.
 - ⌘ La dimensión vertical es el eje del tiempo (hacia abajo).
 - ⌘ La dimensión horizontal muestra los clasificadores con papel que representan objetos.
Rectángulo con línea punteada.
- ⌘ Mensajes en orden (flechas entre líneas de clasificadores)



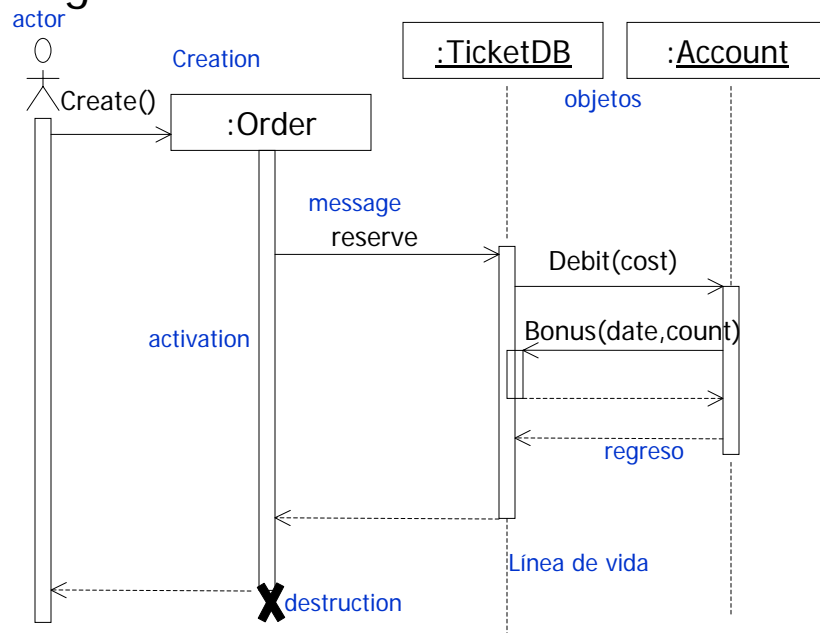
Activación

⌘ Activación es la ejecución de un procedimiento, incluyendo el tiempo de espera para procedimientos anidados. Aparece como línea doble.

⌘ Llamada: flecha al inicio de la activación.

⌘ Una llamada recursiva o anidada aparece como otra línea doble encimada.

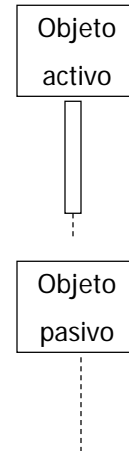
Diagrama de secuencia con activaciones



Objeto activo

⌘ Objeto activo: tiene la raíz de una pila de activaciones, tiene su propio hilo de control que se ejecuta en paralelo con los de otros objetos activos.

⌘ Objetos llamados por un objeto activo son objetos pasivos, estos reciben el control cuando son llamados.



Ejemplos con orden complejo

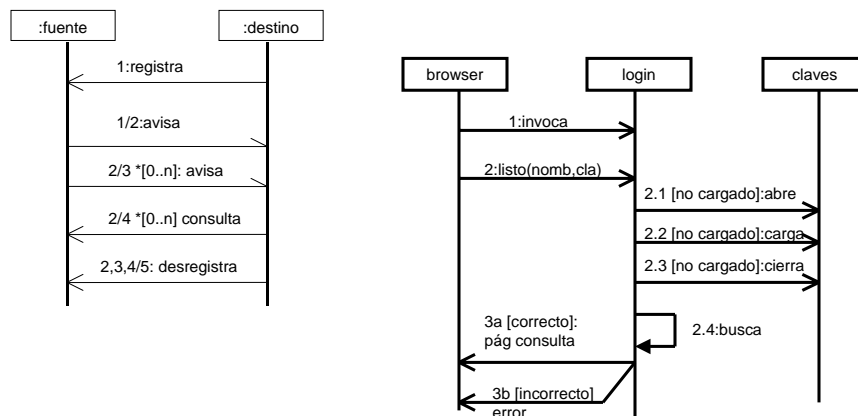
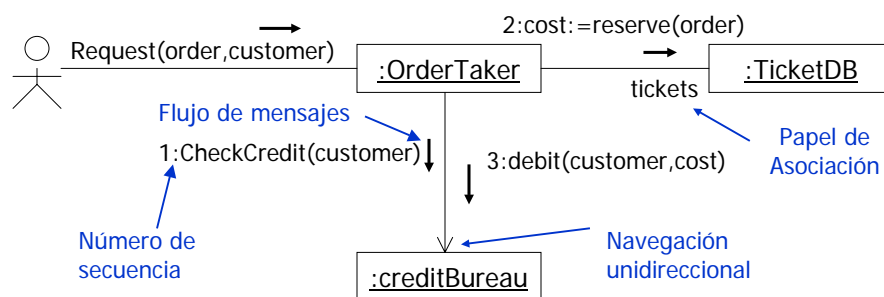


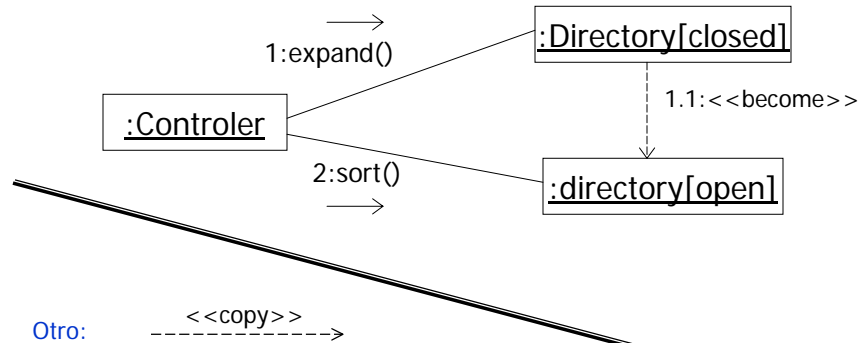
Diagrama de Colaboración

- ⌘ Diagrama de clases pero con énfasis en papeles, tal como ocurrirán en una interacción dada;
- ⌘ Solo se representan los objetos que intervienen en una interacción
- ⌘ Ligas pueden ser temporales, mensajes y parámetros.
- ⌘ Mensajes numerados para indicar orden

Diagrama de Colaboración



Flujos que cambian objetos



Vista de máquina de estados

Vista de máquina de estados

- ⌘ Describe comportamiento dinámico de objetos sobre el tiempo.
- ⌘ Vista aislada; se comunica con el exterior recibiendo eventos y enviando respuestas.
- ⌘ Usa máquinas de estados (caso de autómatas finitos; d. de Harel).
- ⌘ Se aplica también a casos de uso, operaciones, colaboraciones y otros

Máquina (diagrama) de estados

- ⌘ Modela comportamiento de clase
- ⌘ grafo de estados y transiciones
- ⌘ modela todas las posibles historias de la clase
- ⌘ buena para especificar con precisión
- ⌘ mala para entender funcionamiento global

Máquina (diagrama) de estados

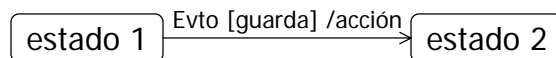
- ⌘ Estado: conjunto de valores de un objeto que producen una misma respuesta (cualitativamente)
- ⌘ Estado: período de espera por la ocurrencia de un evento
- ⌘ Estado: período durante el cual se realiza una actividad

estado

entry/acción exit/acción

Máquina (diagrama) de estados

- ⌘ Transición: respuesta de una clase, en un estado, a la ocurrencia de evento
- ⌘ Formada por:
 - ☒ estado origen
 - ☒ evento causante,
 - ☒ guarda,
 - ☒ acción,
 - ☒ estado destino



Máquina (diagrama) de estados

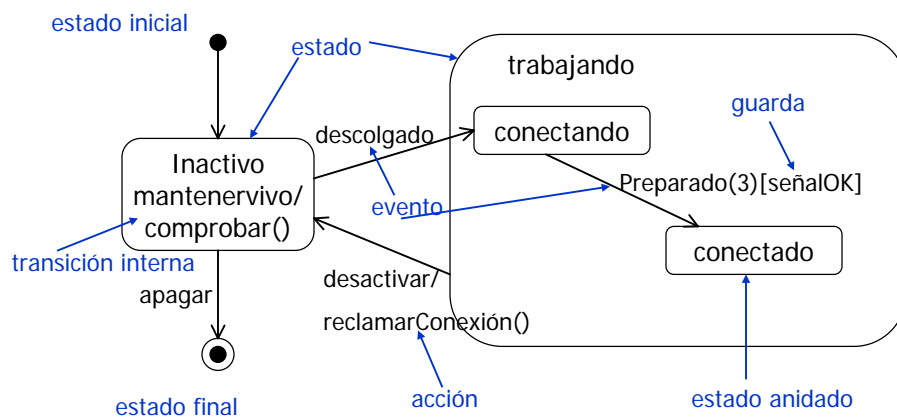
⌘ Evento:

- ☒ llamada, método(parámetros)
- ☒ señal, señal(parámetros)
- ☒ tiempo transcurrido, after(tiempo)
- ☒ cambio de valores when(condición)

⌘ Acción:

- ☒ llamada, señal, cambio de valores
- ☒ crear, destruir, terminar, regresar, asignar

Máquina de estados





Vista de actividades

Vista de actividades



- ⌘ Modela flujo de trabajo o de cómputos; similar a diagrama de flujo
- ⌘ Caso particular de diagrama de estados
- ⌘ Supone no hay interferencia de eventos externos; transiciones por terminación
- ⌘ Puede usarse para modelar actividades de empresa o de grupos de clases

Vista de actividades

⌘ Partes que puede tener:

- ☒ estados de actividad
- ☒ estados de acción (atómicos)
- ☒ transiciones
- ☒ ramificaciones
- ☒ sincronizaciones
- ☒ objetos en flujo
- ☒ carriles (cuando hay varias clases)

Diagrama de actividades

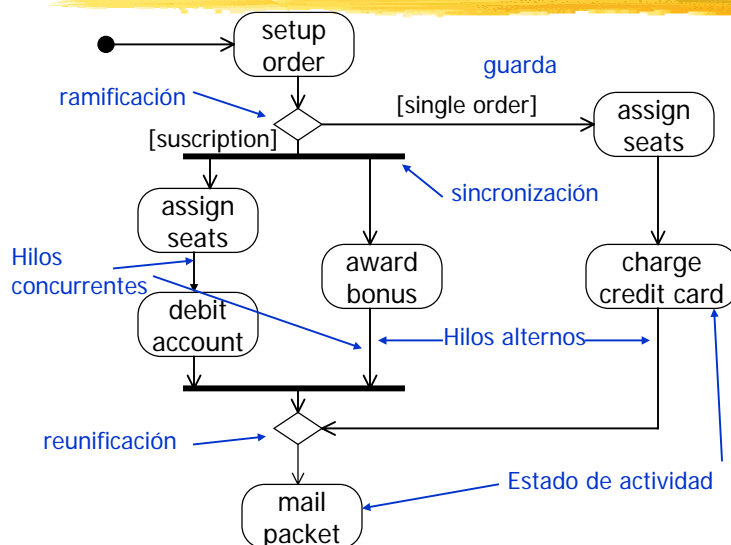
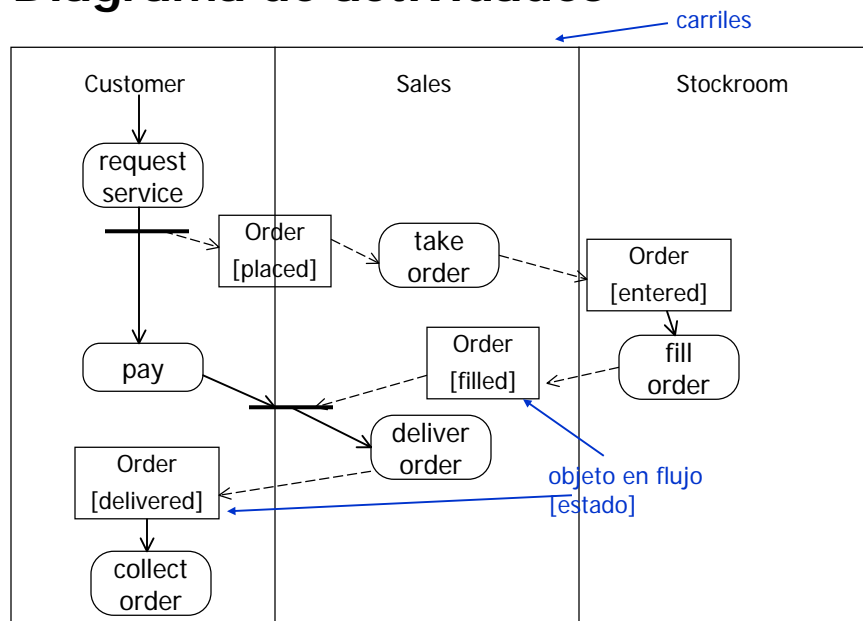


Diagrama de actividades



Vistas físicas:

- ♠ implementación
- ♠ despliegue

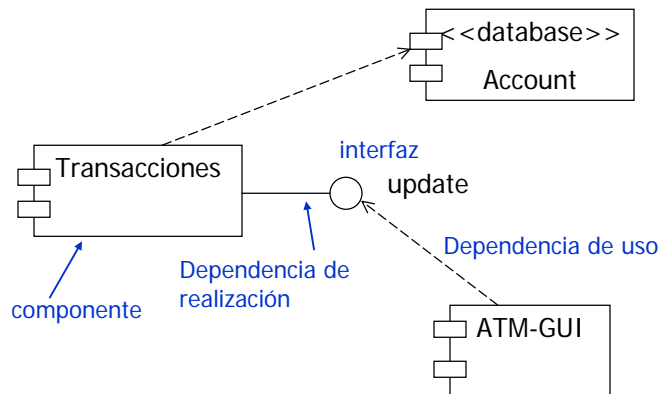
Vistas físicas

- ⌘ Aspectos de implantación (diferente de diseño)
- ⌘ Implementación: cómo se empaca el sistema en piezas reutilizables (componentes)
- ⌘ Despliegue: cómo se organizan recursos de cómputo en ejecución (componentes en nodos)

Implementación

- ⌘ Componente: unidad física de implantación con interfaces bien definidas; será usado como parte de un sistema.
- ⌘ Interfaz: conjunto de operaciones; **no se refiere a interfaz de usuario.**
- ⌘ Dependencias: muestra relaciones entre componentes
- ⌘ Se concreta en diagrama de componentes

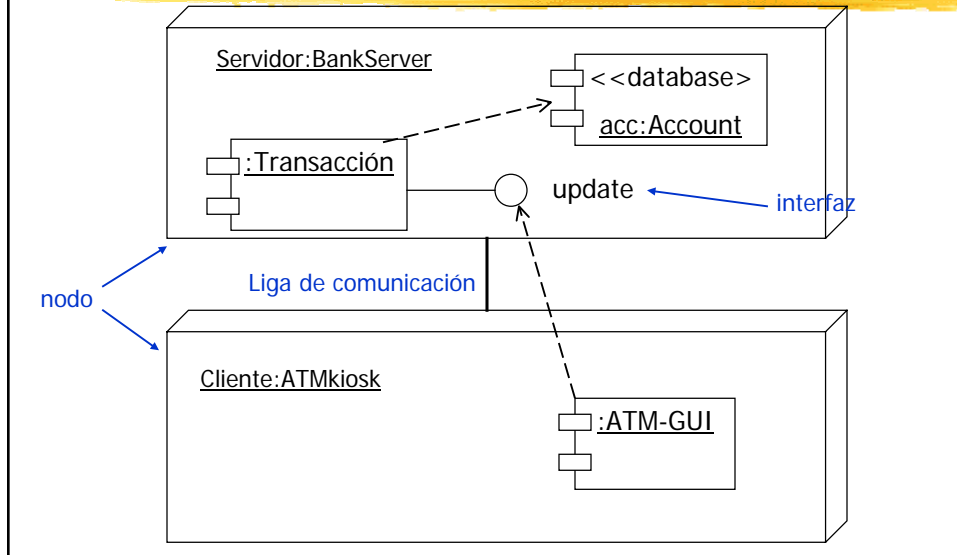
Diagrama de componentes



Despliegue

- ⌘ **Nodo**: objeto físico en tiempo de ejecución que representa recursos computacionales.
- ⌘ Los nodos pueden tener varios estereotipos para distinguir los diferentes tipos de recursos, tal como CPU, red o dispositivos.
- ⌘ Un nodo es mostrado como un cubo con nombre (y clasificación). Asociaciones representan comunicación.
- ⌘ Se incluyen componentes que residen en él.

Diagrama de despliegue

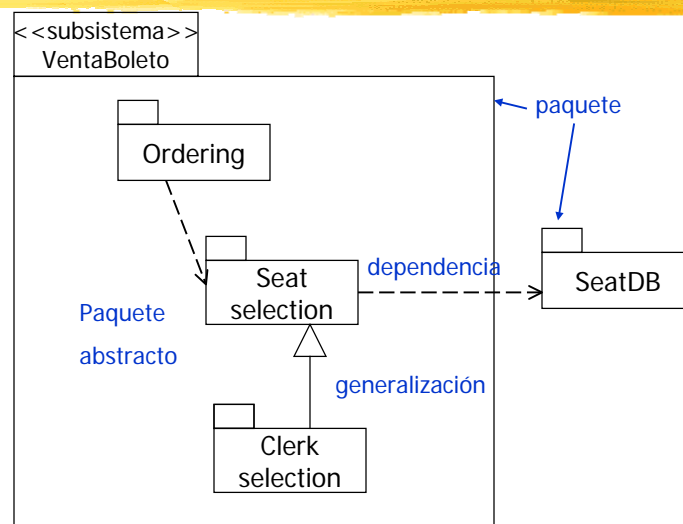


Vista de administración
del modelo

Administración del Modelo

- ⌘ Para facilitar manejo del modelo en sus etapas y partes
- ⌘ Para facilitar control de versiones
- ⌘ Se concreta en Paquetes
- ⌘ Paquete: Caja con oreja para nombre
- ⌘ Pueden llevar estereotipo <<subsistema>>
- ⌘ Todo elemento del modelo es parte de algún paquete
- ⌘ Existen dependencias entre paquetes

Diagrama de paquetes



Mecanismos de extensión

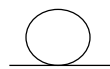
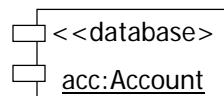
Mecanismos de extensión

- ⌘ Para necesidades especiales, se puede modificar sin rehacer todo UML
- ⌘ Las herramientas los manejan, aún sin conocer su semántica
- ⌘ Tres mecanismos, por separado o conjuntamente:
 - ☒ estereotipos
 - ☒ restricciones
 - ☒ valores etiquetados

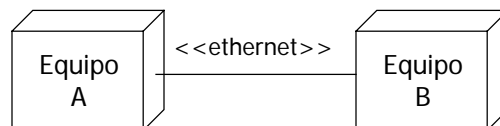
Estereotipo

- ⌘ concepto adicional, definido en el propio modelo.
- ⌘ Basado en un elemento existente.
- ⌘ Se anota como: <<estereotipo>>
- ⌘ Se aplica a clases, asociaciones, etc.
- ⌘ Puede incluir valores etiquetados, un icono especial y restricciones que serán comunes a todas sus instancias

Ejemplos de estereotipo



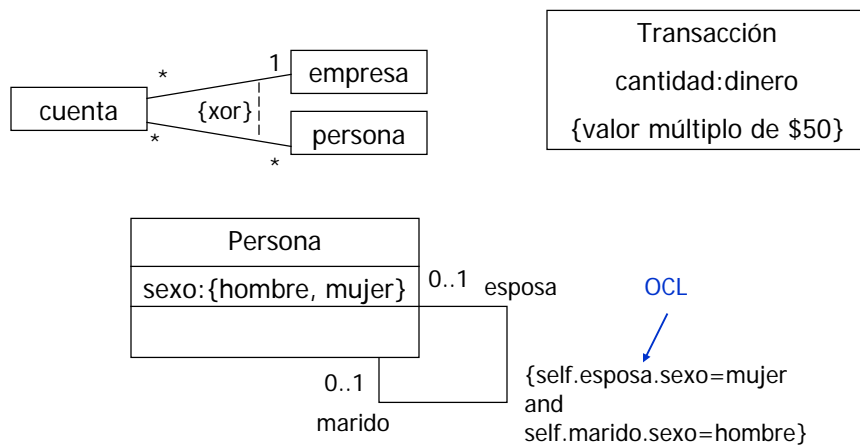
despachador



Restricción

- ⌘ Restricciones semánticas representadas como texto
- ⌘ Lenguaje libre, pero se supone conocido; notación de lógica, conjuntos, lenguaje de programación, lenguaje natural o bien OCL (object constraint language)
- ⌘ se anotan entre llaves {restricción}

Ejemplos de restricción



Valor etiquetado

- ⌘ Definición de valores constantes
- ⌘ Cuerda con etiqueta (tag), signo = y cuerda con valor
- ⌘ Cuando van asociados con estereotipos se define el conjunto de valores posibles
- ⌘ ejemplo: **versión**=1.3.2
- ⌘ ejemplo: **nombre**= Abraham González

Información detallada

- ⌘ Libros de Rumbaugh, Jacobson y Booch
- ⌘ Otros libros sobre UML
- ⌘ Estándar de UML en www.omg.org
- ⌘ Archivos depositados provisionalmente en 148.226.81.2/~ingesoft