Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

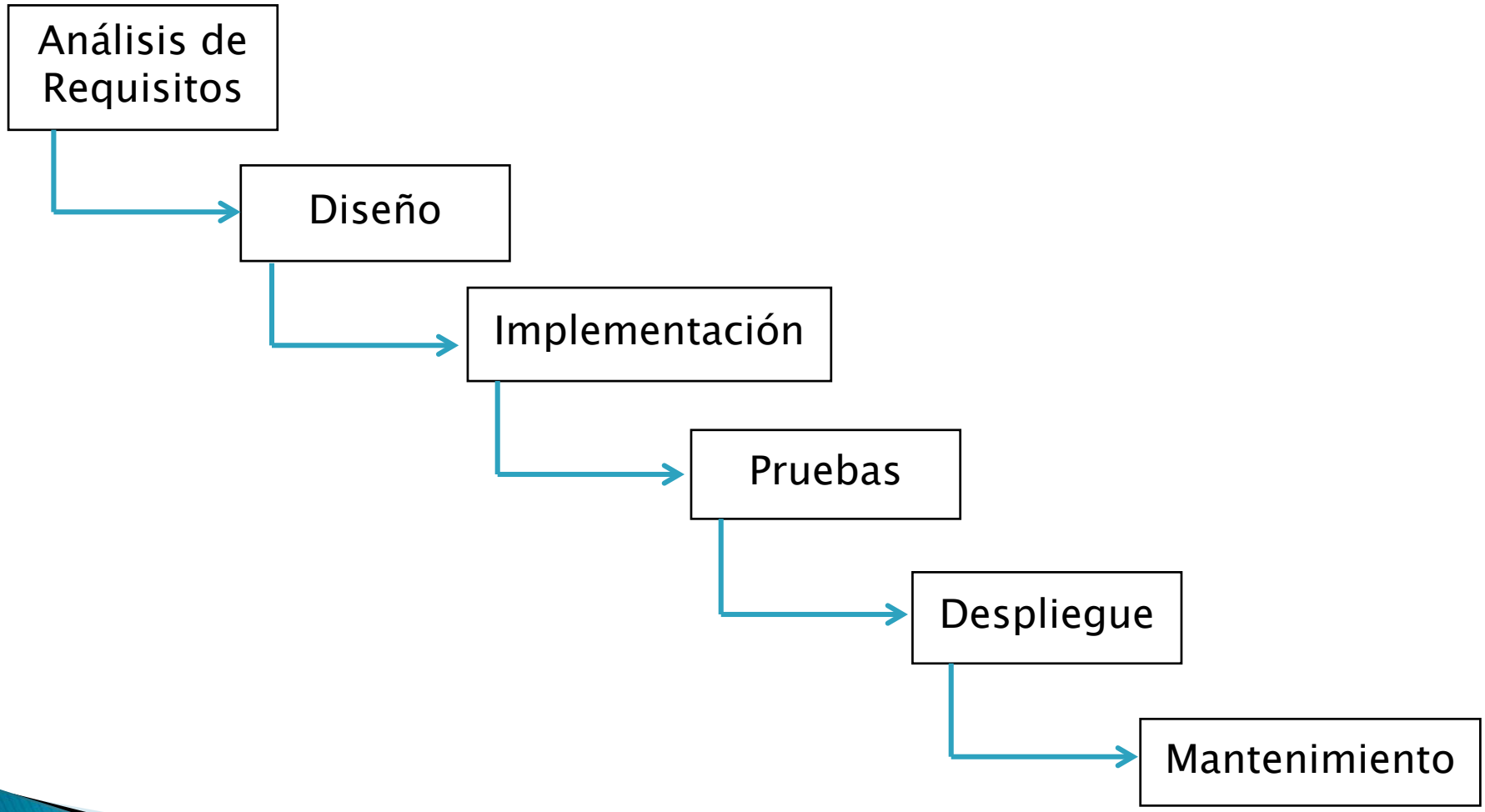
Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

Fases de desarrollo del software



Modelado

- Ideas, relaciones, decisiones
- **Abstracción**, grandes rasgos, postergar detalles
- Forma unívoca de comunicación
- Diferentes modelos para diferentes vistas

Abstracción

- Un modelo de sistema se crea con diferentes niveles de abstracción
  - Comenzando en los niveles superiores y añadiendo más niveles con más detalle
- La abstracción tiene que ver con
  - Mostrar sólo la información que sea relevante en cada momento
  - Ocultar detalles para no confundir la visión global

UML (I)

- UML (Unified Modeling Language) es un lenguaje de modelado de software
  - Proporciona un vocabulario y reglas para crear modelos software
  - Suficientemente expresivo para cubrir distintas vistas de la arquitectura del software a lo largo del ciclo de vida
  - Mayor nivel de abstracción que un lenguaje de programación

UML (II)

- UML es un lenguaje para visualizar los elementos de un gran sistema software, facilitando
  - La comunicación entre los participantes (incluidas herramientas) en el desarrollo
  - La comprensión de las soluciones (notación gráfica)
  - El mantenimiento de las soluciones conceptuales a lo largo del tiempo (documentación)

UML (III)

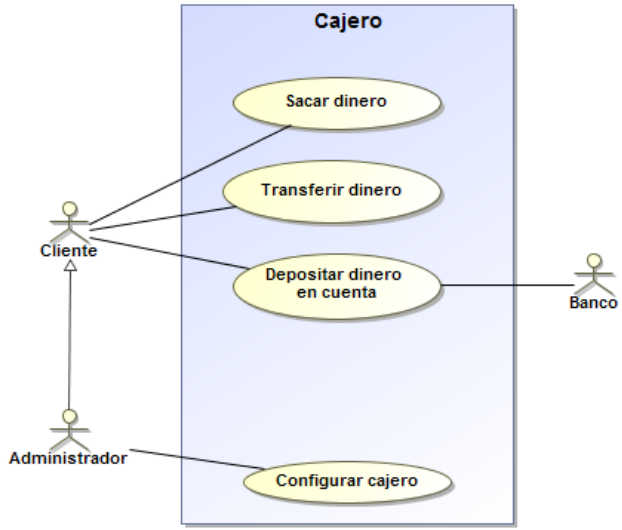
- UML es un lenguaje para especificar software
  - Se pueden construir modelos precisos, no ambiguos y completos
  - Cubre las decisiones de análisis, diseño e implementación

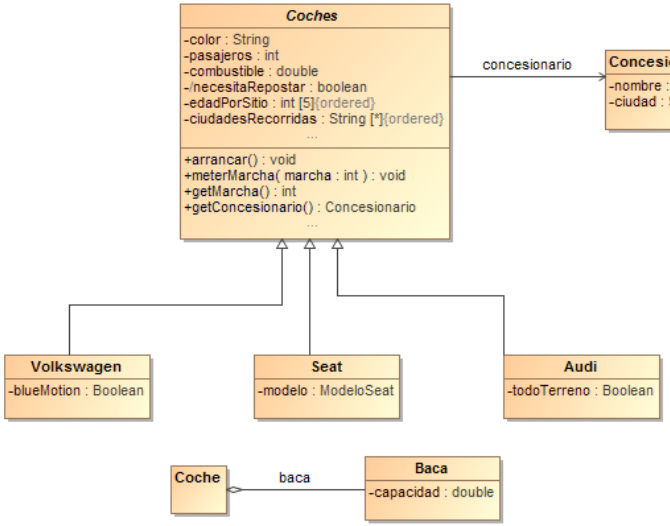
UML (IV)

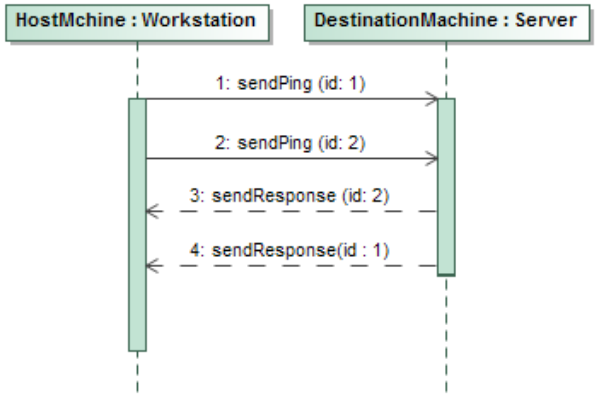
- UML es un lenguaje para construir software
  - No es un lenguaje de programación visual, pero sus modelos se pueden conectar de forma directa a una gran variedad de ellos
  - Correspondencias entre UML y lenguajes: Java, C++,...
  - Ingeniería directa: generación de código
  - Ingeniería inversa: reconstrucción de modelos
- UML es un lenguaje para documentar
  - requisitos, arquitectura, diseño, código fuente, pruebas

Modelado con UML

- Los modelos UML se basan en diagramas
- Los diagramas UML contienen conceptos y relaciones entre ellos
- Diferentes diagramas para diferentes vistas

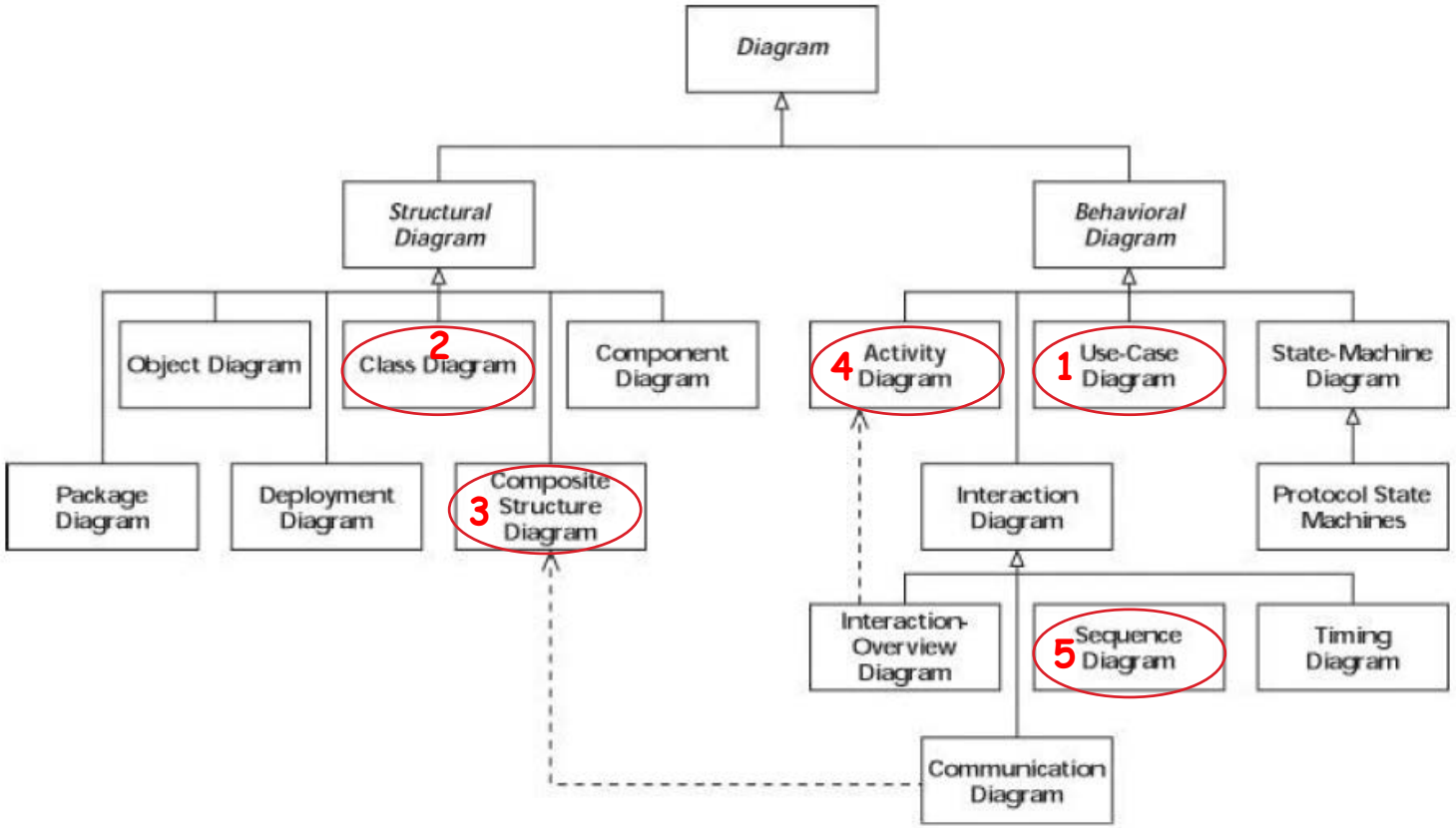






Diagramas en UML

- Diagramas estructurales y de comportamiento



Cómo trabajar con UML (I)

- Usar todos (y únicamente) los diagramas y características necesarios
- Tener en cuenta los destinatarios
- Tener en cuenta el objetivo, qué se quiere modelar

Cómo trabajar con UML (II)

- Crear componentes
- Separar lo que hace un componente de cómo lo hace
- Proporcionar un mecanismo común de comunicación
- Permitir la presencia de componentes en el entorno y las relaciones entre ellos

Cómo trabajar con UML (III)

- UML es una notación, no una metodología
- Hay múltiples metodologías con actividades comunes como:
  - Toma de requisitos
  - Análisis
  - Diseño
  - Implementación
  - Pruebas
- Diferentes familias de metodologías
  - En cascada
  - En espiral
  - Iterativos
  - Ágiles

Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

UML : Casos de Uso
(Diagrama de Comportamiento)

Casos de Uso

- Los casos de uso describen el comportamiento de un sistema desde el **punto de vista del usuario** según acciones y reacciones
- Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno
- Son descripciones de la funcionalidad del sistema independientes de la implementación
  - Describen ***qué*** hace el sistema, no ***cómo*** lo hace
- Particionan el conjunto de necesidades atendiendo a la categoría de usuario que participan en el mismo
- Está basado en el lenguaje natural

Elementos de un diagrama de casos de uso

- Un diagrama de casos de uso es un grafo constituido por
  - Actores 
  - Casos de uso 

  - Relaciones entre elementos 

Actores (I)

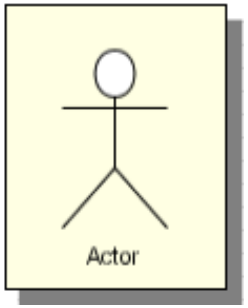
Un Actor:

- Representa un tipo de usuario
- Es una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema de la misma forma

✓ Los actores se representan con dibujos simplificados de personas, llamados en inglés "stick man" (hombres de palo). La notación puede adecuarse al contexto.

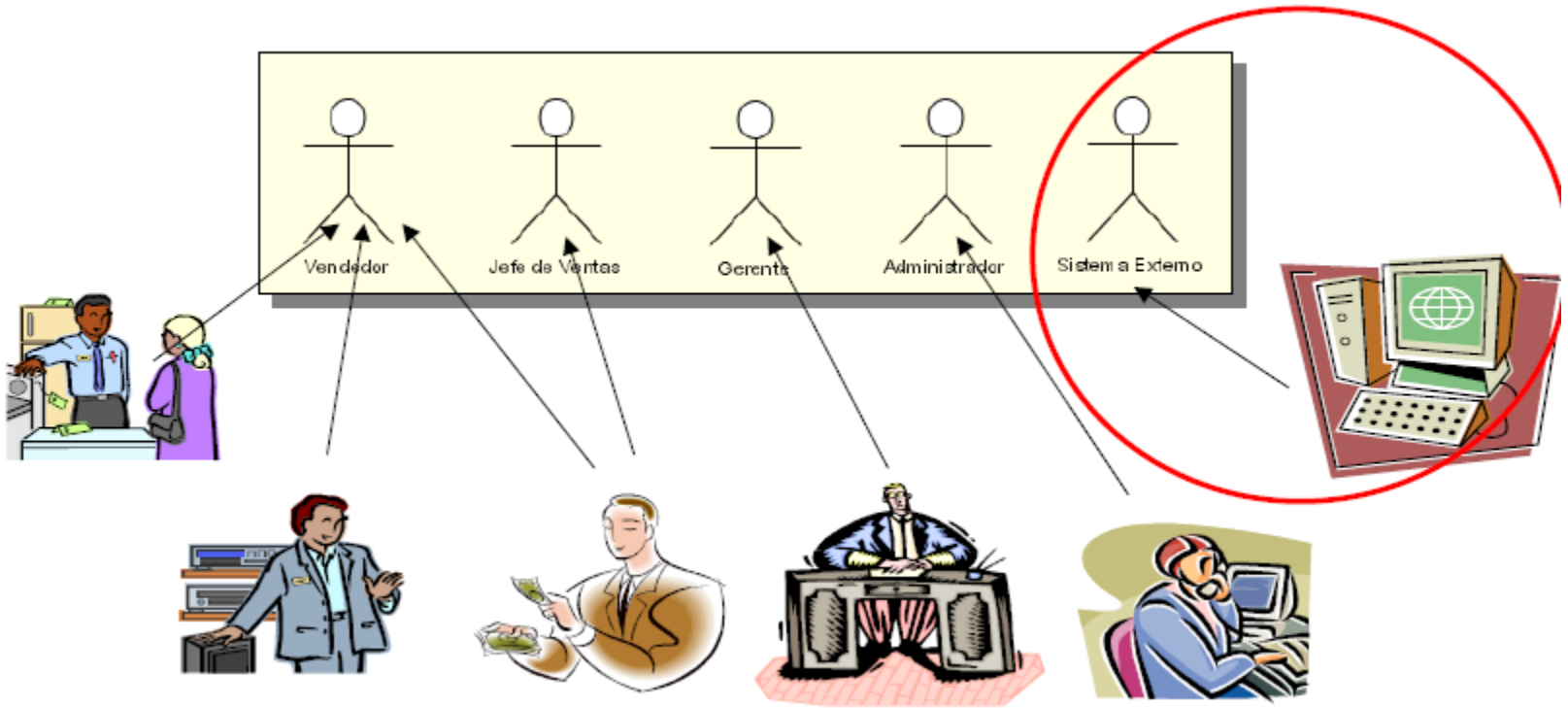
✓ La misma persona física puede interpretar varios papeles como actores distintos.

✓ El nombre del actor describe el papel desempeñado.



Actores (II)

- La misma persona física puede interpretar varios papeles como actores distintos



Elementos de un diagrama de casos de uso

- Un diagrama de casos de uso es un grafo constituido por
  - Actores 
  - **Casos de uso** 

  - Relaciones entre elementos 

Caso de uso

Definición

- Un caso de uso describe una **interacción** coherente y con un objetivo de un **actor** con un sistema
- al comienzo del cuál hay un disparador del dominio (*trigger*),
- y al final del cual hay un resultado definido con un valor de dominio

- Se representa con una elipse

Ejemplo: buscaminas (I)

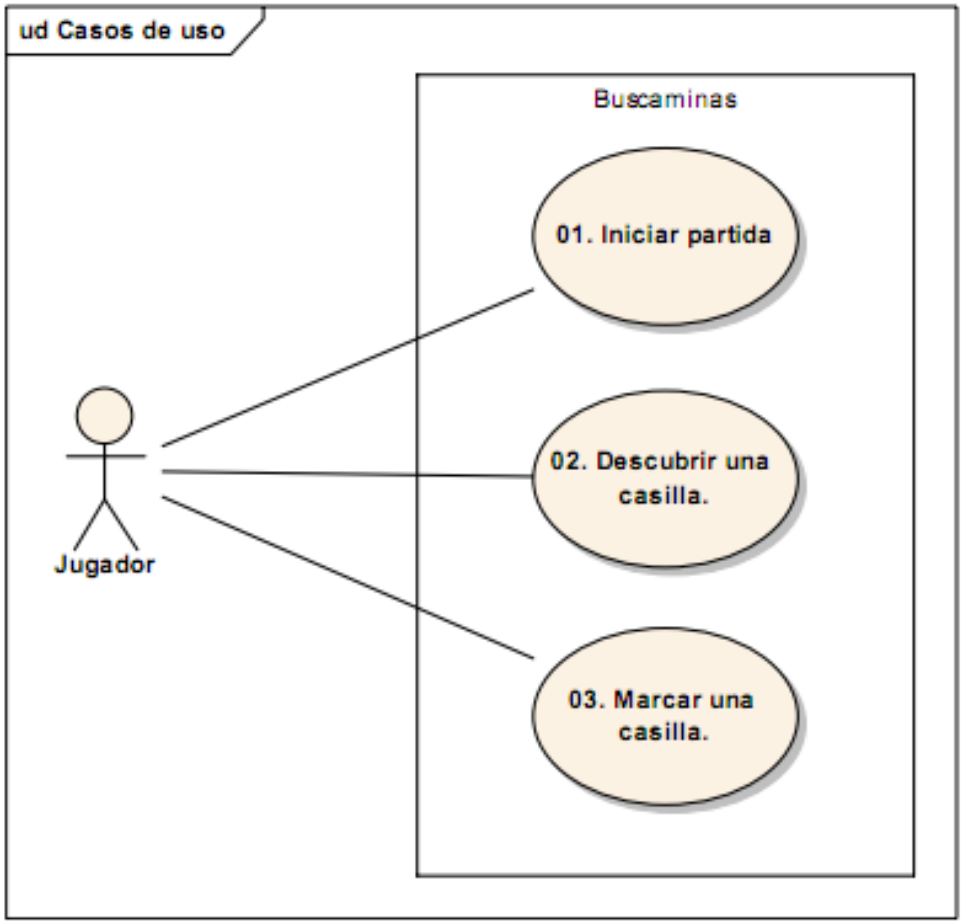
¿Qué casos de uso identificamos?

- Iniciar una nueva partida.
- Descubrir una casilla.
- Marcar una casilla.

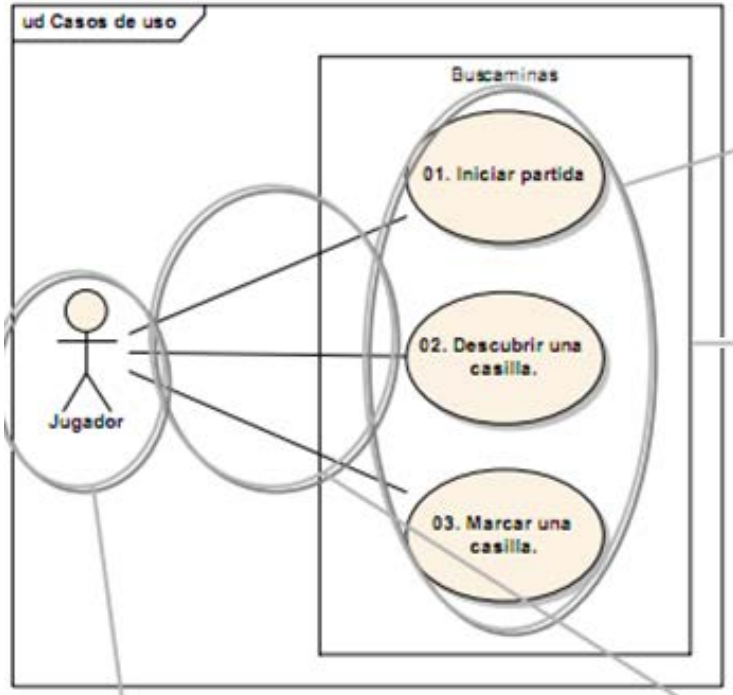
¿Quién realiza estos casos de uso?

- El jugador.

Ejemplo: buscaminas (II)



Ejemplo: buscaminas (III)



Caso de Uso: interacción entre actores y el sistema que produce un resultado observable de valor para un actor.

Límite del sistema: agrupa casos de uso dentro de un mismo sistema.

Actor: alguien o algo externo al sistema que interactúa con él desempeñando un rol.

Un caso de uso ***siempre*** es iniciado por un actor externo.

Asociación: la participación de un actor es necesaria para realizar el caso de uso.

Elementos de un diagrama de casos de uso

- Un diagrama de casos de uso es un grafo constituido por
  - Actores 
  - Casos de uso 

  - **Relaciones entre elementos**

Relaciones entre elementos

- Relaciones de comunicación entre actores y casos de uso
- Relaciones de Generalización/Especialización entre actores
- Relaciones entre casos de uso
  - Generalización/Especialización
  - Inclusión
  - Extensión

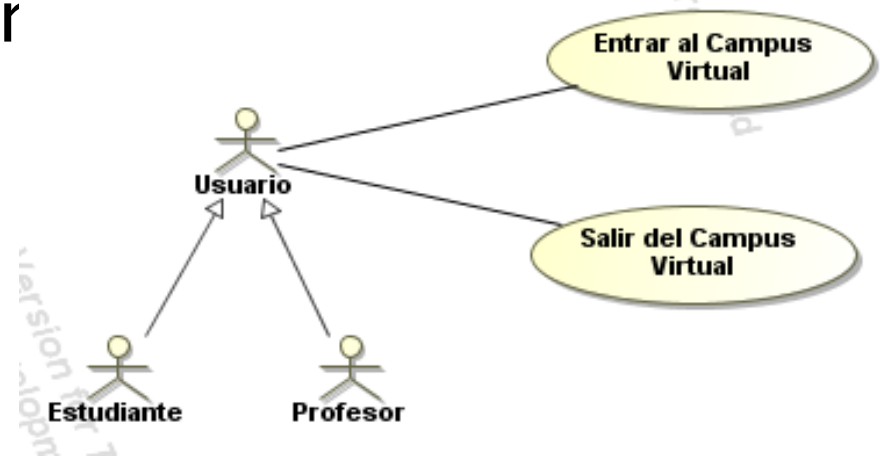
Relación Actor – Caso de uso

- Es una relación de **comunicación**
- Representa comunicación en uno o dos sentidos
- Se representa gráficamente mediante una línea continua

Relación Actor – Actor

- Relación de **generalización**: se pueden definir categorías generales de actores y especializarlos

- Esta relación se representa gráficamente mediante una flecha con la cabeza hueca dirigida al actor r

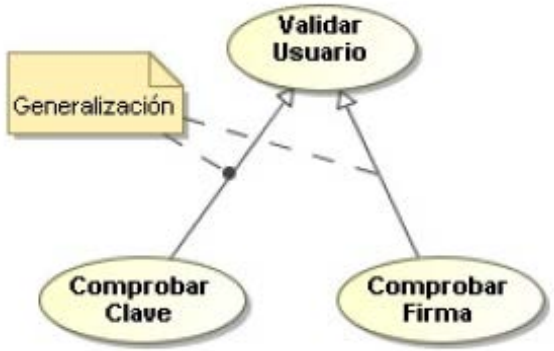


Relaciones. Generalización

- Un caso de uso (hijo) hereda el comportamiento y significado de otro caso de uso (padre)

- El caso de uso hijo pude añadir comportamiento o bien redefinir el del padre. Puede colocarse donde aparezca el padre
  - Ejemplo: un caso de uso Validar usuario puede tener como hijos los casos de uso Comprobar Clave y Comprobar firma

- Se representa gráficamente con una flecha hueca dirigida al caso de uso padre



Relaciones. Inclusión

- Una relación de inclusión (include) denota que un caso de uso está incluido en otro
  - Se da cuando un caso de uso se utiliza por sí mismo y, además, otros casos de uso incluyen siempre esa funcionalidad
  - También aparecen cuando dos o más casos de uso comparten una funcionalidad
  - Denota que un caso de uso SIEMPRE está incluido en otro caso de uso

- Gráficamente se representa mediante una flecha etiquetada con el estereotipo <<usa>> (o <<include>>) que parte del caso de uso base

Relaciones. Extensión (I)

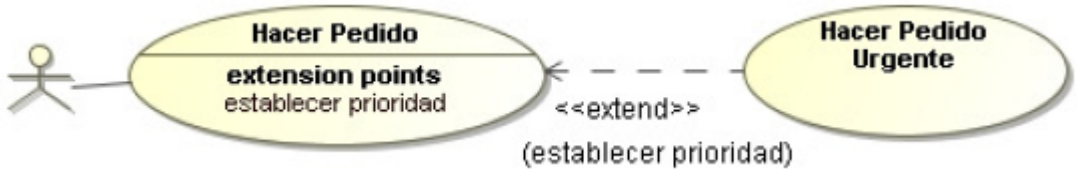
- Una relación de extensión representa un comportamiento OPCIONAL de un caso de uso
  - Se puede indicar mediante un "punto de extensión" (condición) cuando se amplia el caso de uso
  - El caso de uso base, por tanto, bajo ciertas condiciones incorpora el comportamiento de otro caso de uso en un punto concreto de su especificación denominado punto de extensión

- Se representa gráficamente mediante una flecha etiquetada con el estereotipo <<extiende>> (o <<extend>>) que llega al caso de uso base

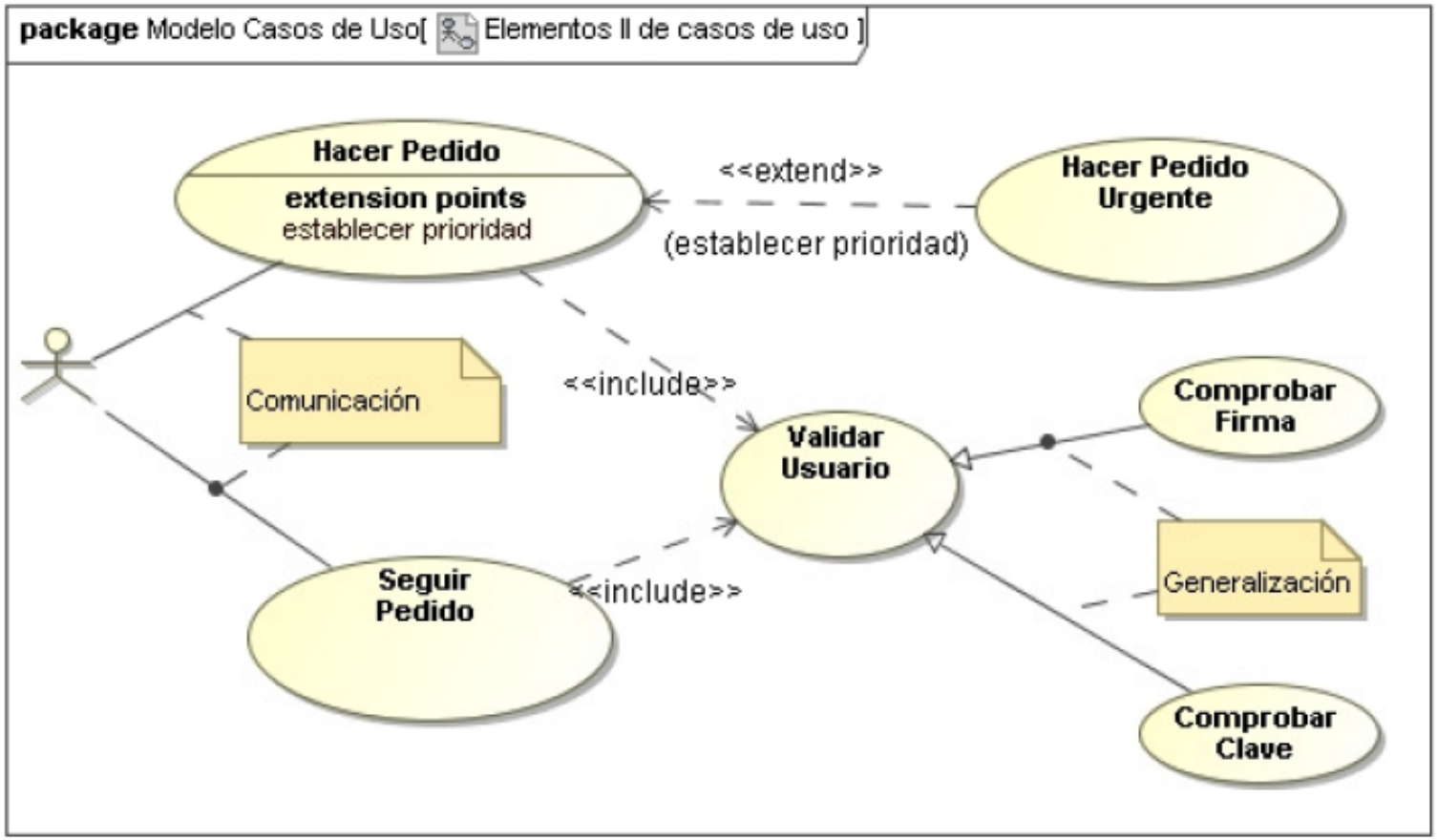
- Un caso de uso base puede tener varias relaciones de extensión y, en consecuencia, varios puntos de extensión

Relaciones. Extensión (II)

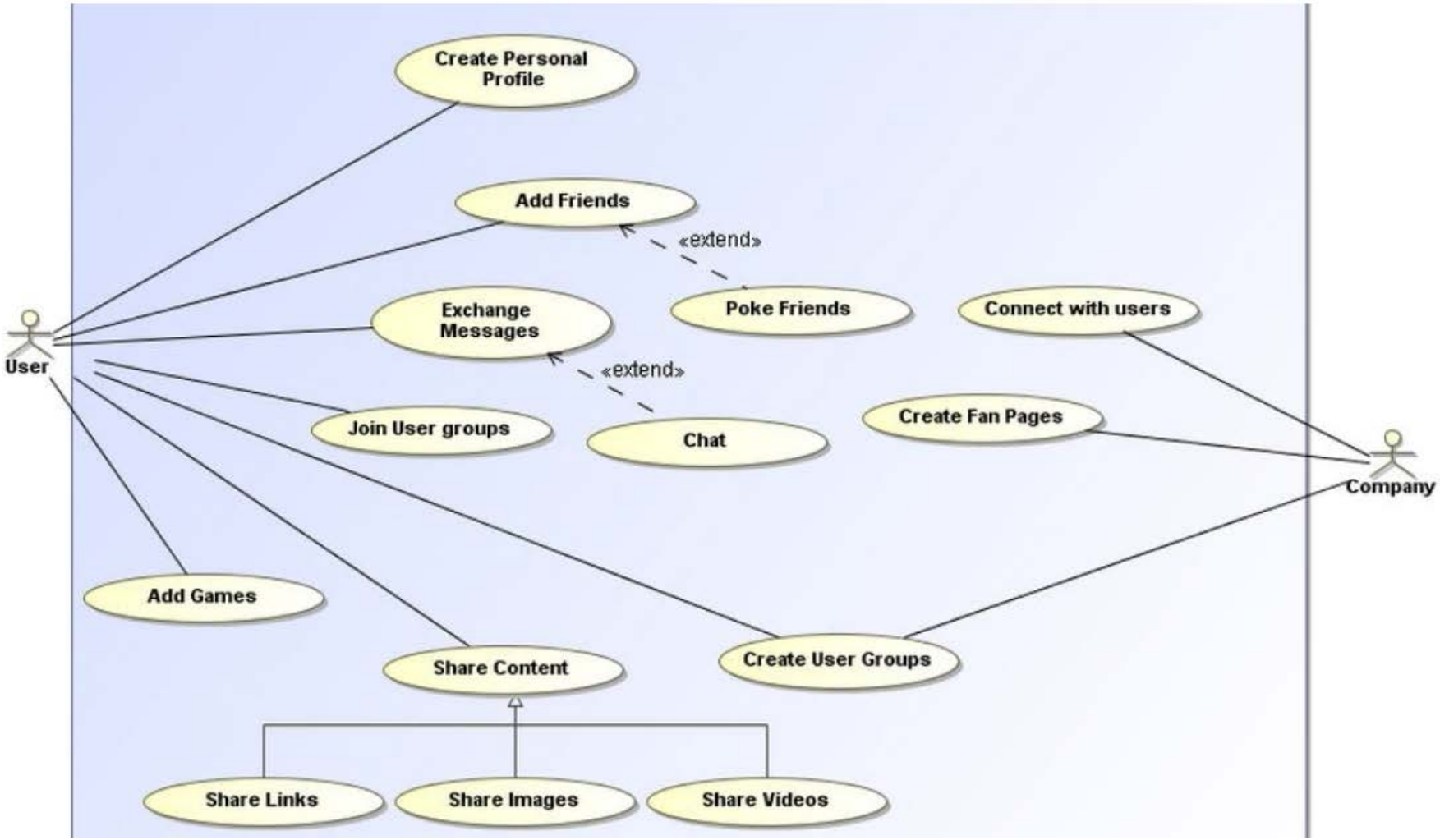
- Ejemplo
  - Consideremos el caso de uso base “Hacer pedido” con una relación de extensión con otro caso de uso “Hacer pedido urgente”
  - La especificación de un escenario normal sería:
    - El cliente solicita hacer un pedido
    - El usuario proporciona los datos del pedido (establecer prioridad)
    - Procesar el pedido



Ejemplo: Pedidos



Ejemplo: Facebook



Ejemplo: retirada de efectivo en cajero

- Tras registrarse en el cajero, un Cliente puede:
  - Sacar dinero
  - Transferir dinero a otras cuentas
  - Depositar dinero en su cuenta
- Si el Cliente, además es un Administrador, puede:
  - Configurar las opciones del cajero
- El Banco también participa, y puede:
  - Depositar dinero en cuentas

Ejemplo: chat

- Se desea desarrollar un sistema de encuentros virtuales (parecido a un chat).
- Cuando se conecta al servidor, un usuario puede entrar o salir de un encuentro.
- Cada encuentro tiene un manager.
- El manager es el usuario que ha planificado el encuentro (el nombre del encuentro, la agenda del encuentro y el moderador del encuentro).
- Cada encuentro puede tener también un moderador designado por el manager.
- La misión del moderador es asignar los turnos de palabra para que los usuarios hablen.
- El moderador también podrá dar por concluido el encuentro en cualquier momento.
- En cualquier momento un usuario puede consultar el estado del sistema, por ejemplo los encuentros planeados y su información.

Ejemplo: juego naves

- Un juego de teléfono móvil donde participan dos jugadores cada uno con su propia terminal.
- Cuando dos jugadores desean jugar, uno de ellos crea una nueva partida y el otro se conecta.
- El objetivo del juego es manejar una nave y disparar al contrario. Si uno de los dos jugadores acierta, la partida termina.
- Si uno de los dos jugadores deja la partida (o se pierde la conexión) la partida termina.

Ejercicio: librería universitaria

- Un prestatario, que puede ser un profesor, estudiante o estudiante de posgrado, puede pedir prestado libros y devolverlos.
  - Cuando se devuelve un libro no hay que hacer nada más
  - Cuando se pide prestado un libro, el prestatario tiene además que identificarse, bien con DNI o con el Carné de Estudiante.
  - Además de pedir prestado el libro, el prestatario puede alquilarlo por tres meses
    - Para alquilar un libro por tres meses hay que pagarlo, bien en efectivo o con tarjeta de crédito.

Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

UML : Diagramas de Actividades

Diagrama de Actividades

- Un diagrama de actividades muestra el **flujo de control** general de un sistema, modelando su **lógica**
  - Modela una **secuencia de acciones y condiciones** tomadas dentro de un proceso

- Se puede utilizar uno para cada operación o caso de uso

Ejemplo

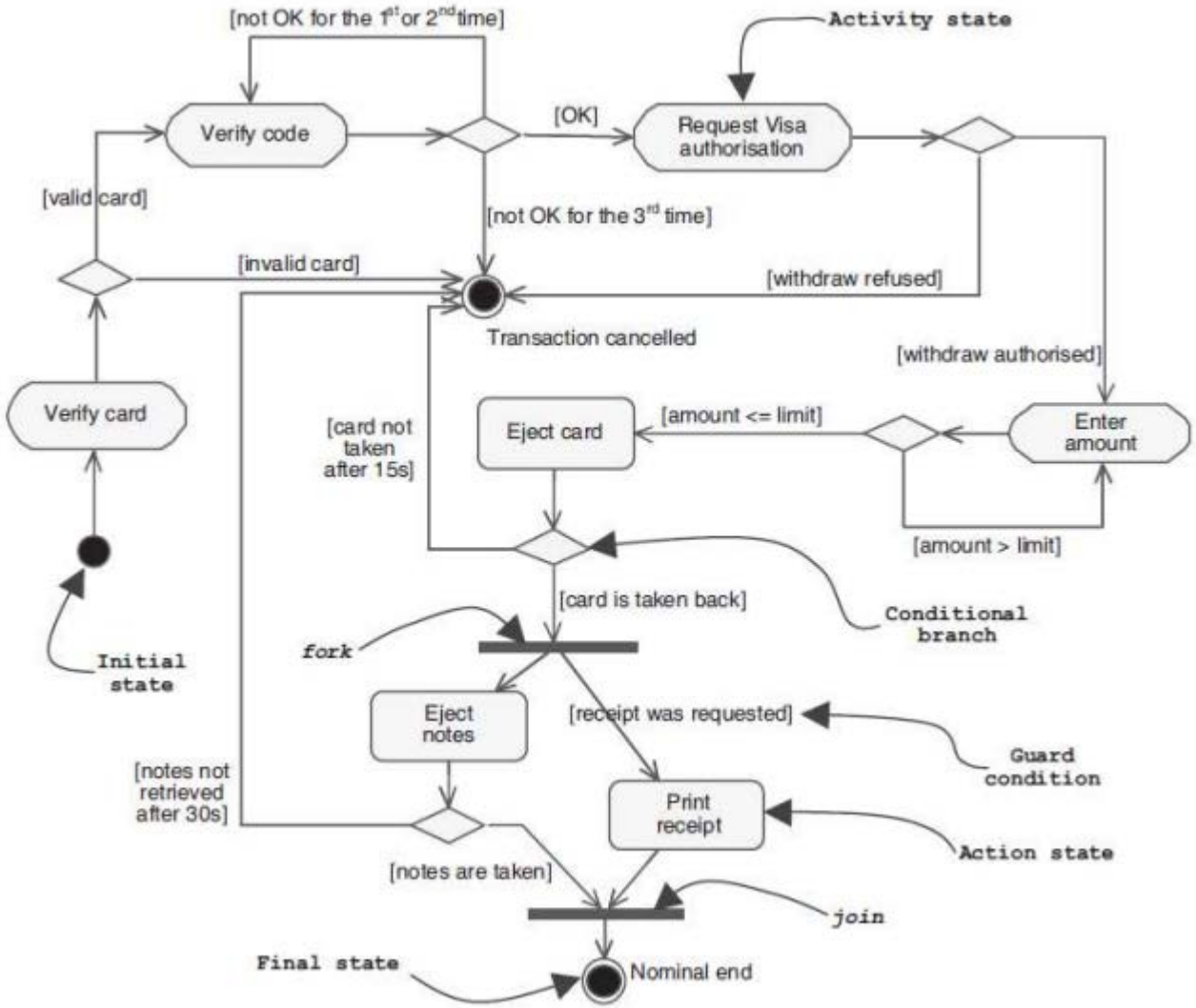

[not OK for the 1st or 2nd time]
Activity state
Verify code
[OK]
Request Visa authorisation
[valid card]
[not OK for the 3rd time]
[invalid card]
[withdraw refused]
Transaction cancelled
[withdraw authorised]
Verify card
[card not taken after 15s]
Eject card
[amount <= limit]
Enter amount
[amount > limit]
[card is taken back]
Conditional branch
fork
Initial state
Eject notes
[receipt was requested]
Guard condition
[notes not retrieved after 30s]
Print receipt
Action state
[notes are taken]
join
Final state
Nominal end

Diagrama de Actividades

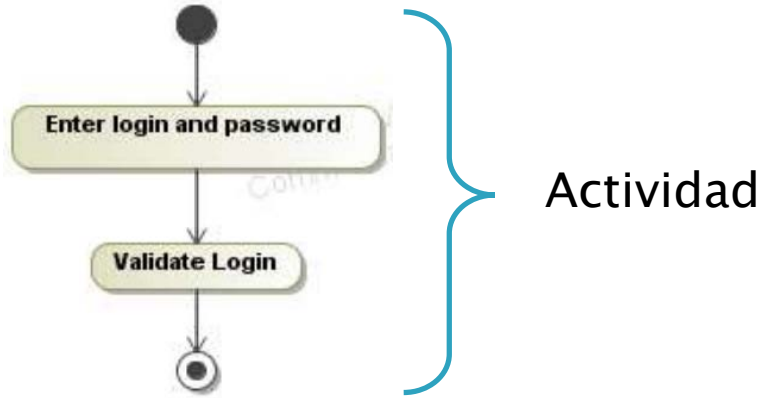
- Cuando usarlos:
  - Para modelar negocios o flujos de trabajo (workflows) inter/intra-sistemas
  - Para modelar el flujo de eventos de un caso de uso
  - Para modelar la implementación de operaciones complejas de una clase
    - Para describir un algoritmo complejo
- Cuando NO usarlos:
  - Para ver cómo se comunican los objetos
    - Usar Diagramas de secuencia o Comunicación
  - Para ver cómo se comporta un objeto
    - Usar Diagramas de máquinas de estado

Elementos

- Actividades
- Acciones
  - Acciones de eventos de aceptación
- Flujos de control
- Inicio y final de actividad
- Nodos objeto
  - Nodos de parámetro de actividad
- Flujos de objetos
  - Pines de entrada y salida
- Precondición y postcondición
- Nodos de control
  - Nodos de decisión/mezcla
  - Nodos fork/join
- Excepciones
- Nodos final de flujo
- Calles (swimlanes)

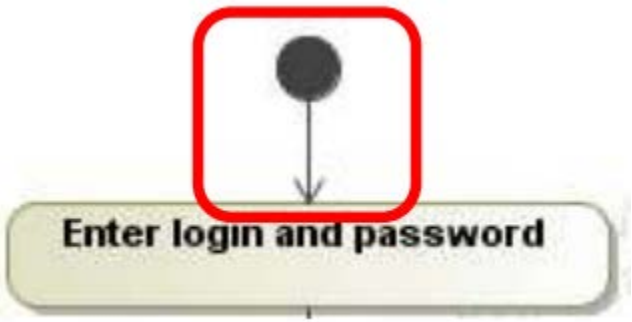
Actividades y transiciones

- Una actividad es un trabajo manual o informatizado que se realiza para producir un resultado
  - Proporcionar datos, validar datos, autorizar acceso, enviar informe, …
- La transición de una actividad a otra se realiza cuando finaliza la actividad origen
- Cada diagrama de actividad representa una actividad



Actividades y transiciones

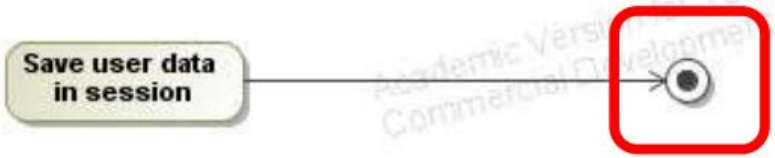
- El nodo inicial muestra el comienzo de la colaboración
- Solo puede tener transiciones de salida
- Habrá un único nodo inicial



Actividades y transiciones

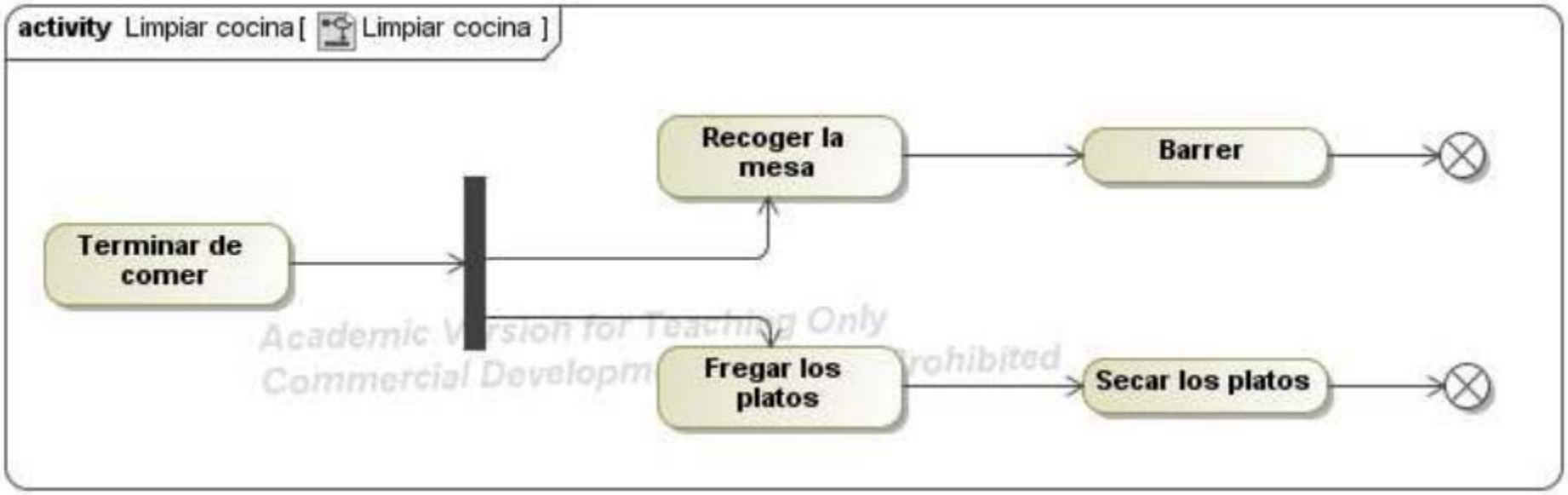
- El nodo final de flujo representa el final de un flujo o secuencia lógica dentro de un diagrama de actividad
  - Permite que los demás flujos continúen su actividad

- El nodo final indica la finalización del diagrama de actividad
  - Interrumpe todos los flujos



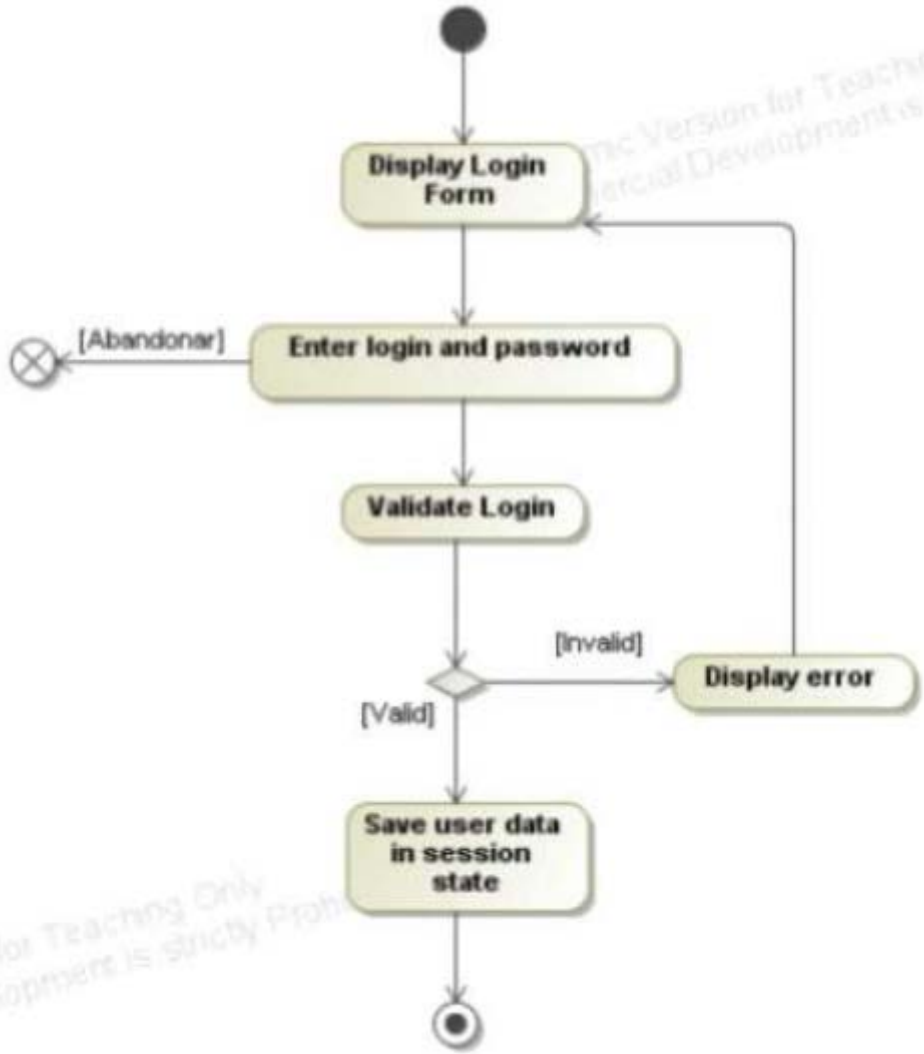
Actividades y transiciones

- Ejemplo:



Actividades y transiciones

- Ejemplo:

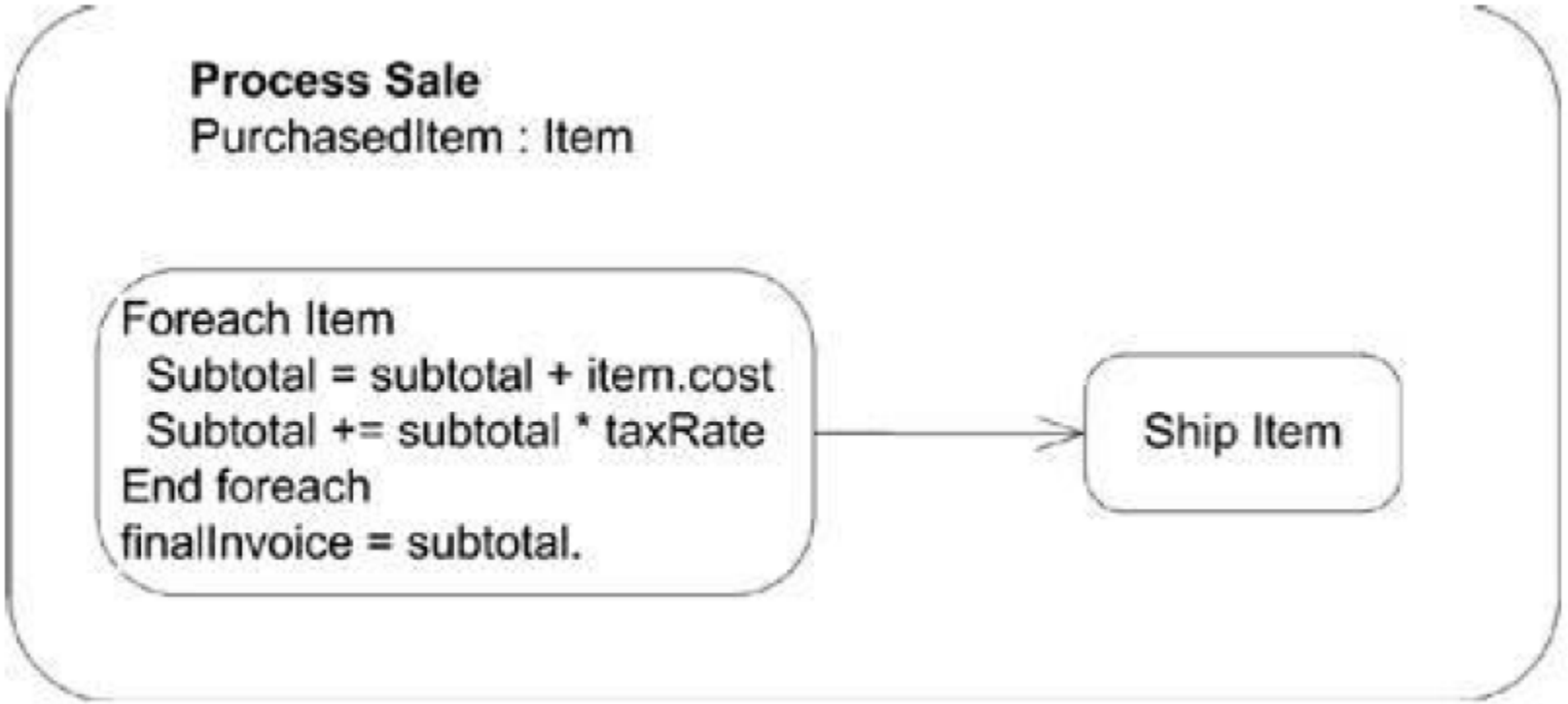


Acción y flujo de control (I)

- Las acciones representan un paso dentro de una actividad
  - Se denotan como un rectángulo redondeado
  - Puede tener un conjunto de acciones de entrada y salida que especifican el flujo de control y de datos desde y hacia otros nodos
- Los flujos de control se representan por la unión entre actividades
  - Una actividad empieza cuando la otra termina

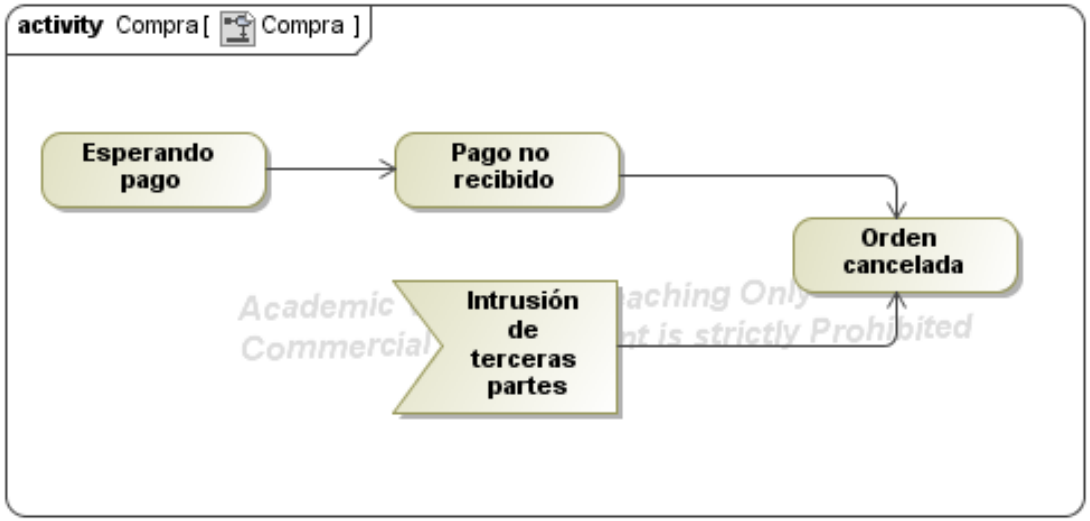
Acción y flujo de control (II)

- Una acción puede mostrar la lógica interna mediante pseudocódigo

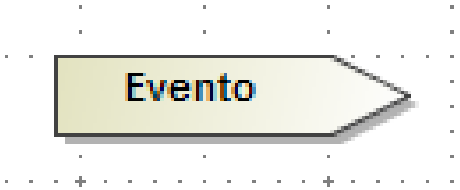


Eventos

- Evento de aceptación: Son acciones que esperan la ocurrencia de un evento
  - El evento que se espera es el texto de la actividad



- Señal de evento: Envían eventos



Nodo objeto y flujo de objeto

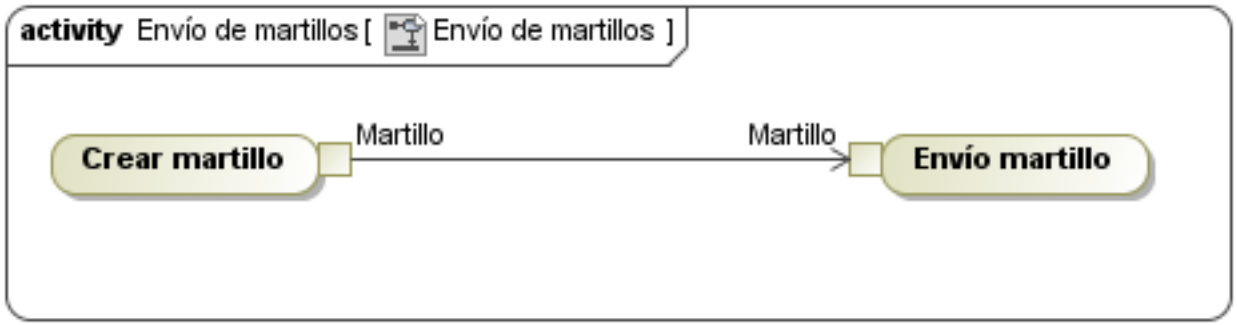
- Nodo que indica que una instancia de una clase puede estar disponible en un punto concreto de la actividad

- El flujo de objetos se utiliza para unir una acción con un nodo objeto y viceversa

Pines de entrada y salida

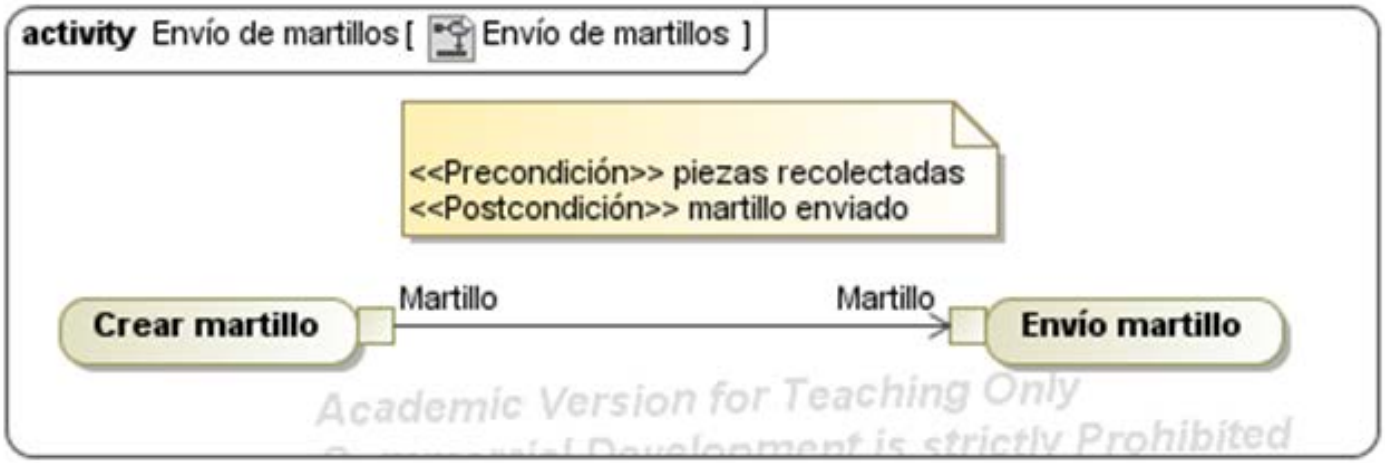
- Los pines de entrada y salida de una acción son una abreviatura gráfica para los nodos objetos
  - Hay uno de entrada y uno de salida

- Los nodos de acción con pines de entrada/salida también se enlazan con flujos de objetos



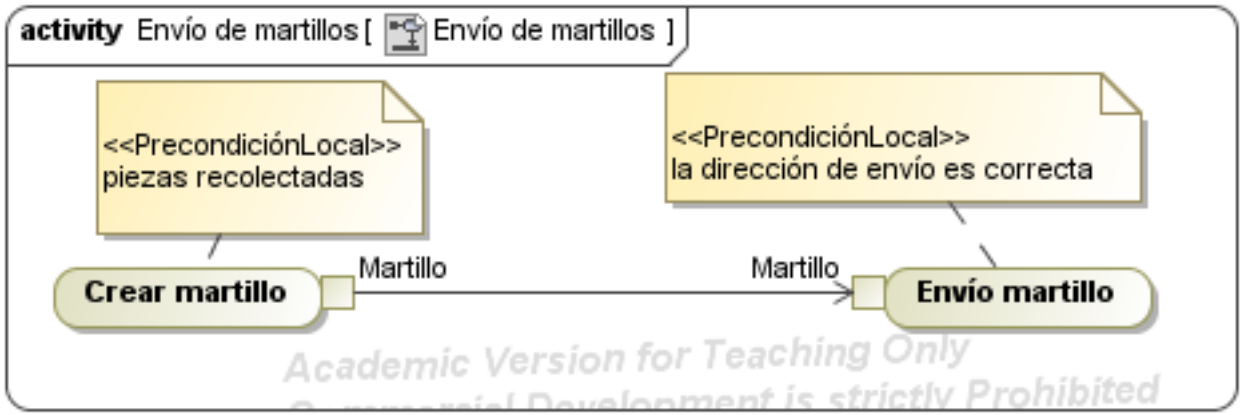
Precondición y postcondición (I)

- Precondición y postcondición de una actividad



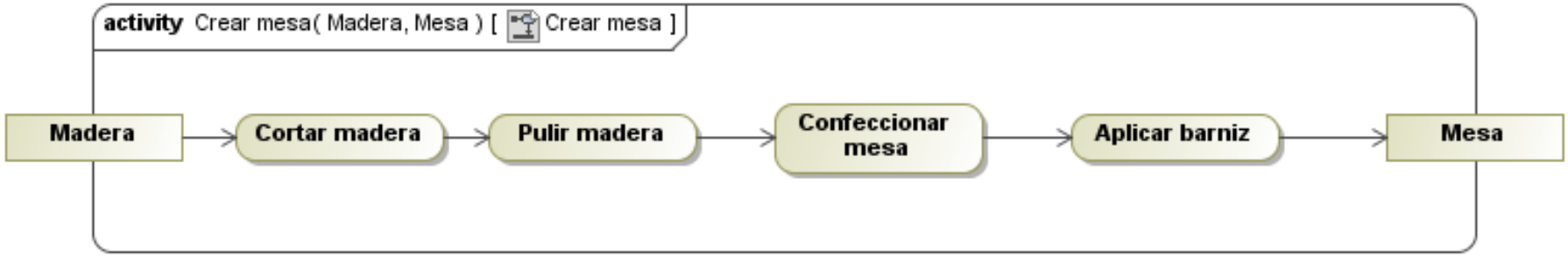
Precondición y postcondición (II)

- Precondición y postcondición locales de una acción



Nodos de parámetros de actividad

- Representan los parámetros de entrada de una actividad y los resultados que produce

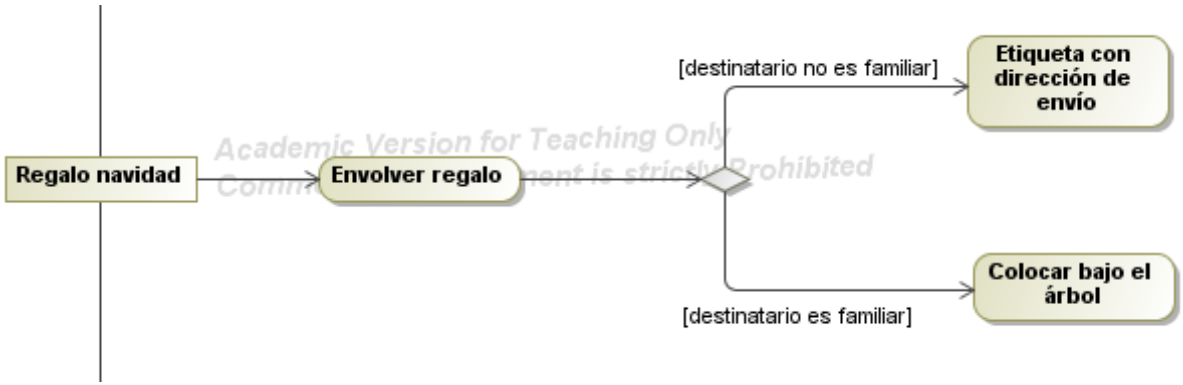


Nodos de control

- Permiten definir decisiones, concurrencia o sincronización
  - Nodos decision/merge
  - Nodos fork/join

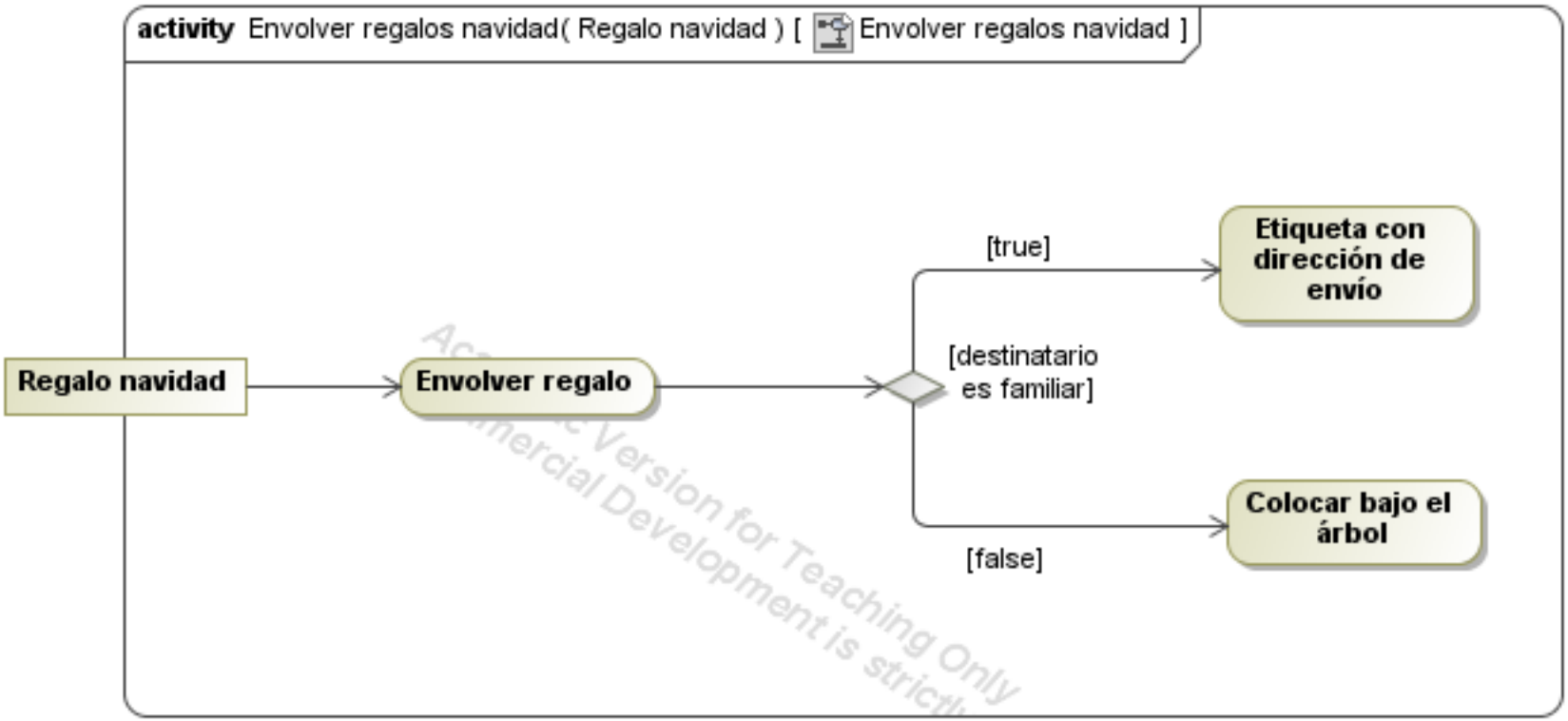
Nodos de decisión (I)

- Un nodo de decisión utiliza un rombo para mostrar donde divergen dos (o más) transiciones alternativas
- Los rombos también se usan para unir de nuevo rutas alternativas
- Permiten escoger entre diferentes flujos en base a guardas (condiciones lógicas)
  - Se sigue uno de los dos caminos



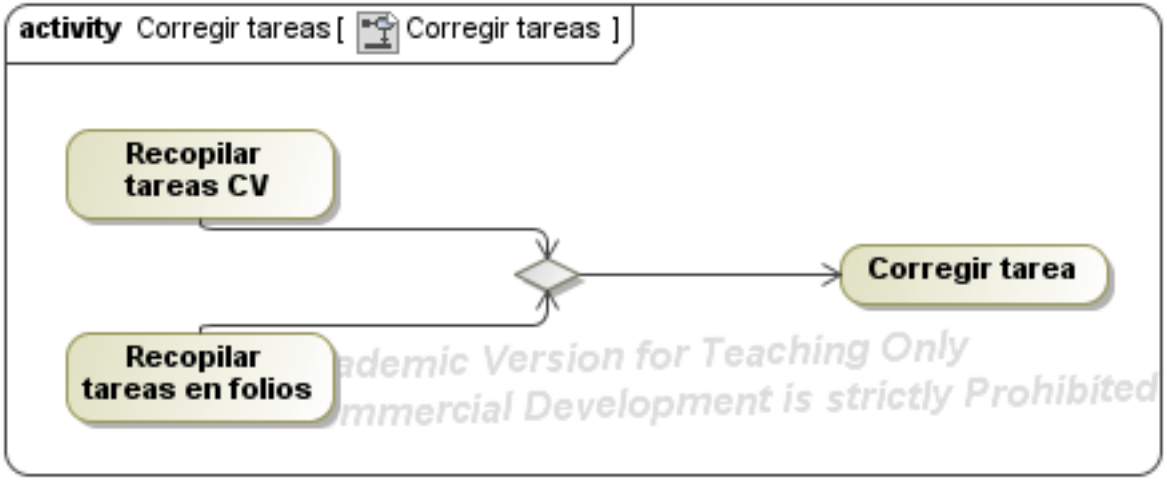
Nodos de decisión (II)

- Otra forma de expresar lo mismo



Nodos de mezcla

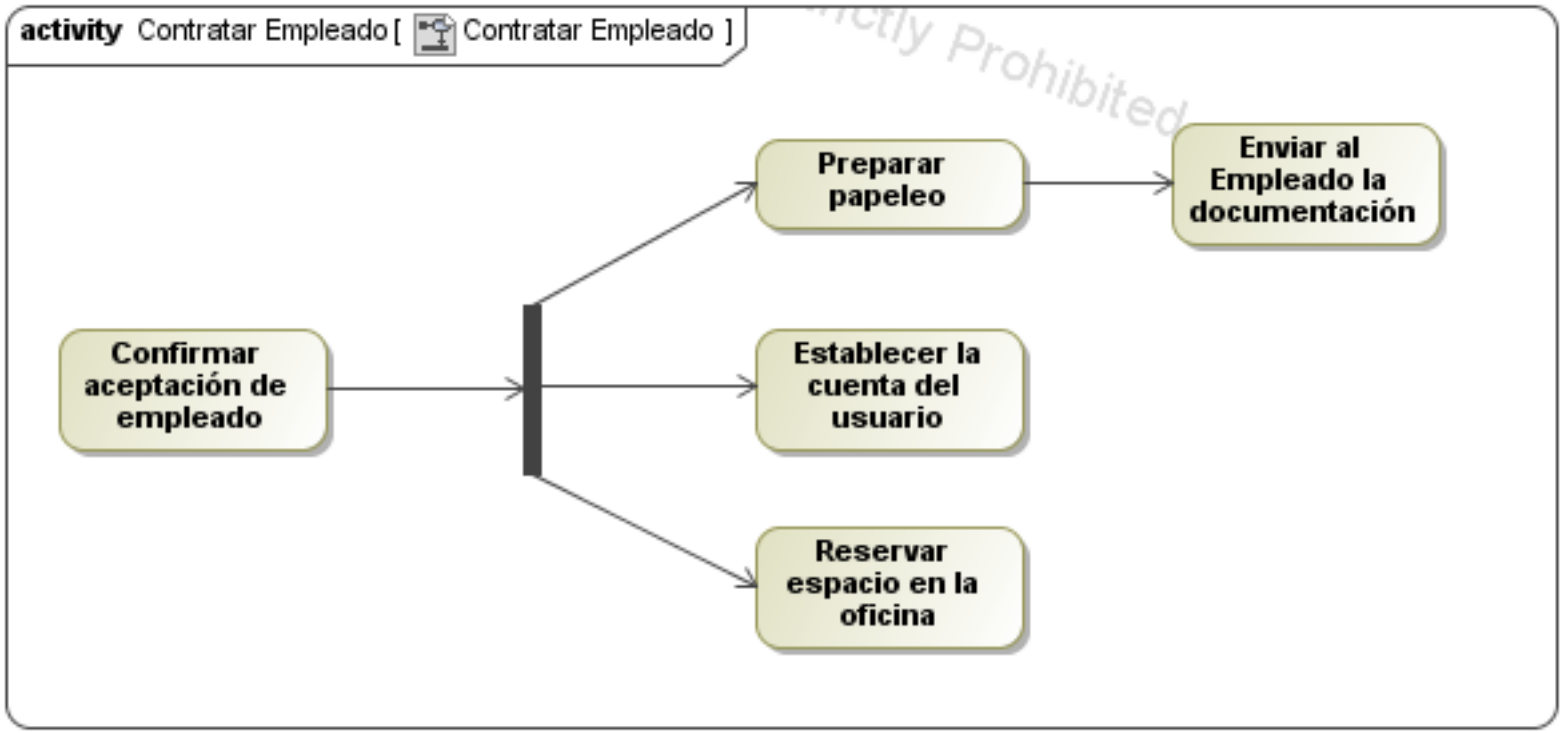
- Unen en un solo flujo varios flujos alternativos



- Sólo es necesario que se ejecute una de las dos acciones para que el flujo continúe

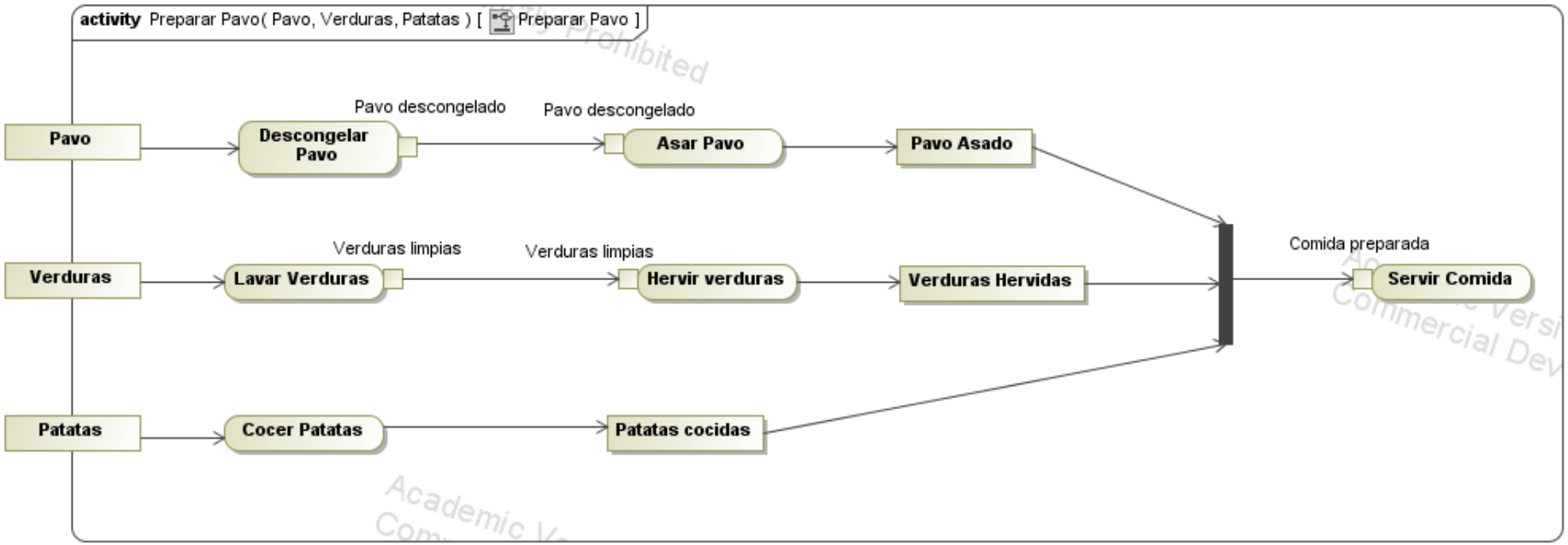
Nodos fork

- Crea varios flujos concurrentes a partir de uno solo



Nodos join (I)

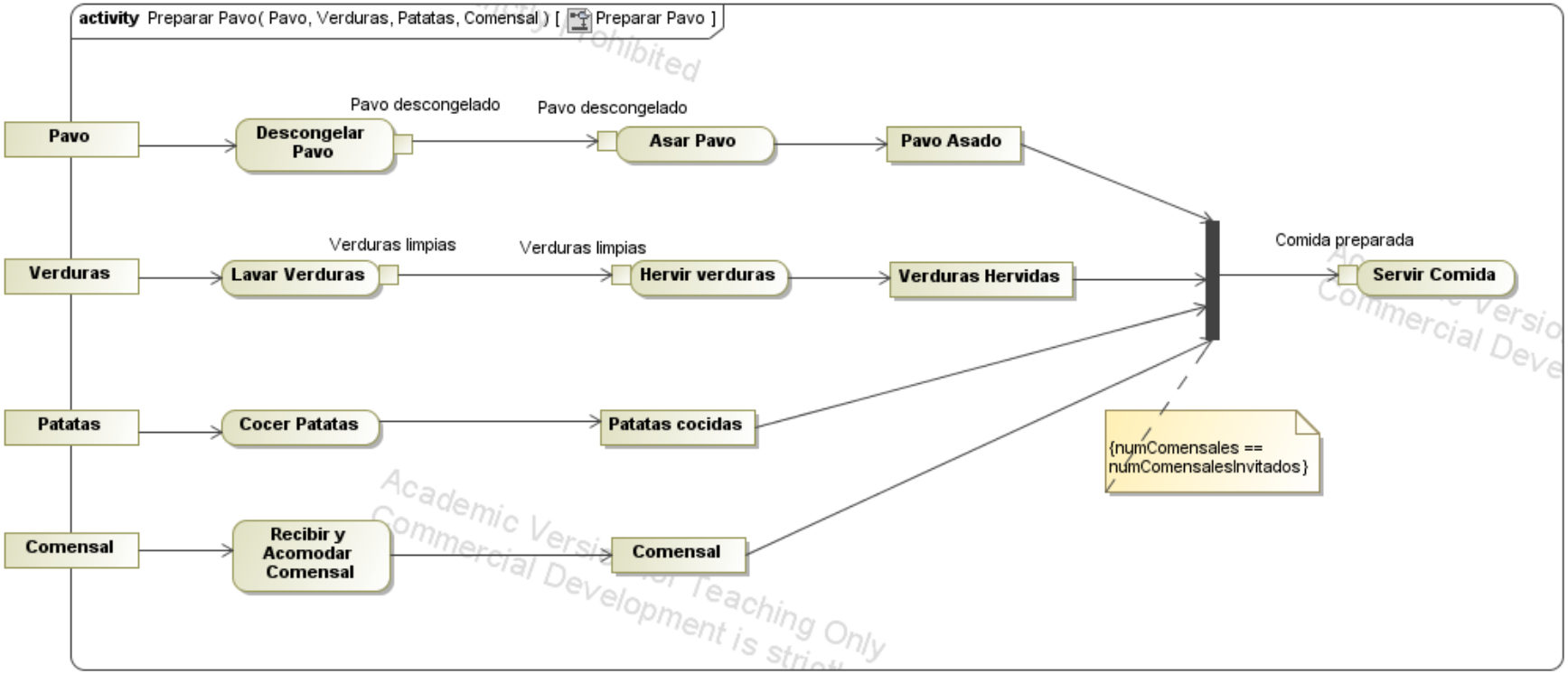
- Sincroniza varios flujos concurrentes y los une en un solo flujo



- Cada rama fluye concurrentemente con las demás hasta que llegan al nodo join. La actividad espera a que todas las ramas hayan llegado al nodo join para ejecutar la actividad “Servir Comida”

Nodos join (II)

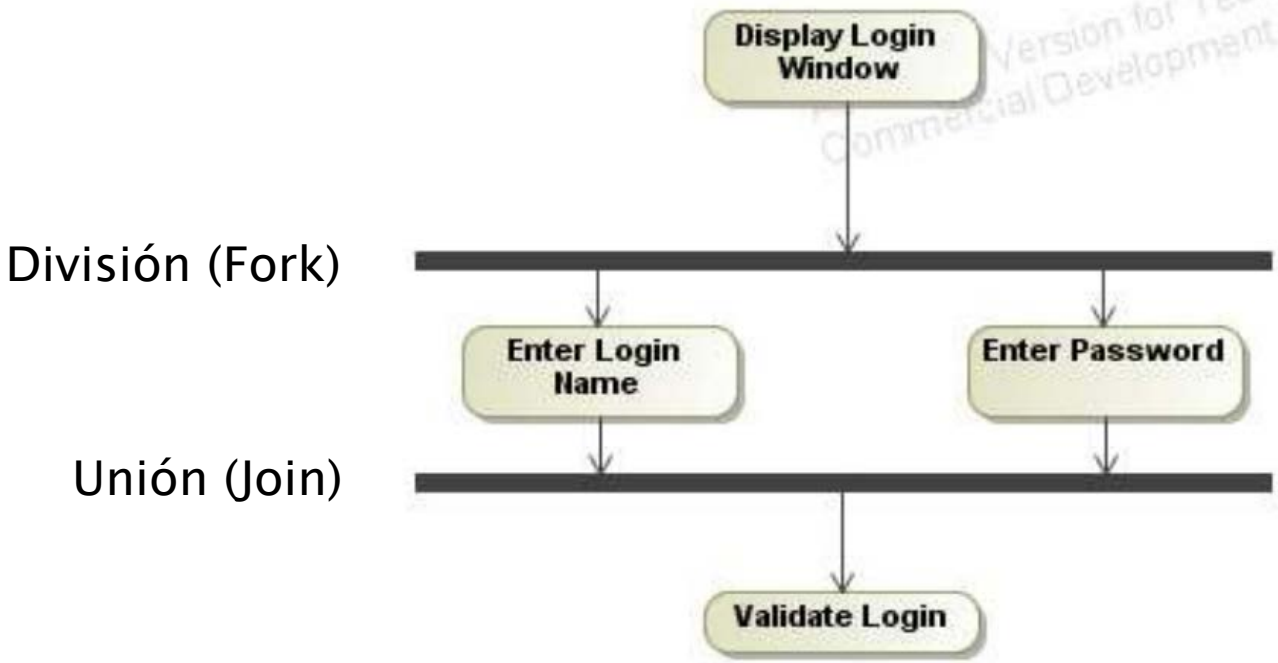
- Se pueden especificar condiciones para la sincronización de los diferentes flujos



- El nodo join no permitirá la ejecución de “Servir Comida” hasta que no hayan llegado todos os comensales

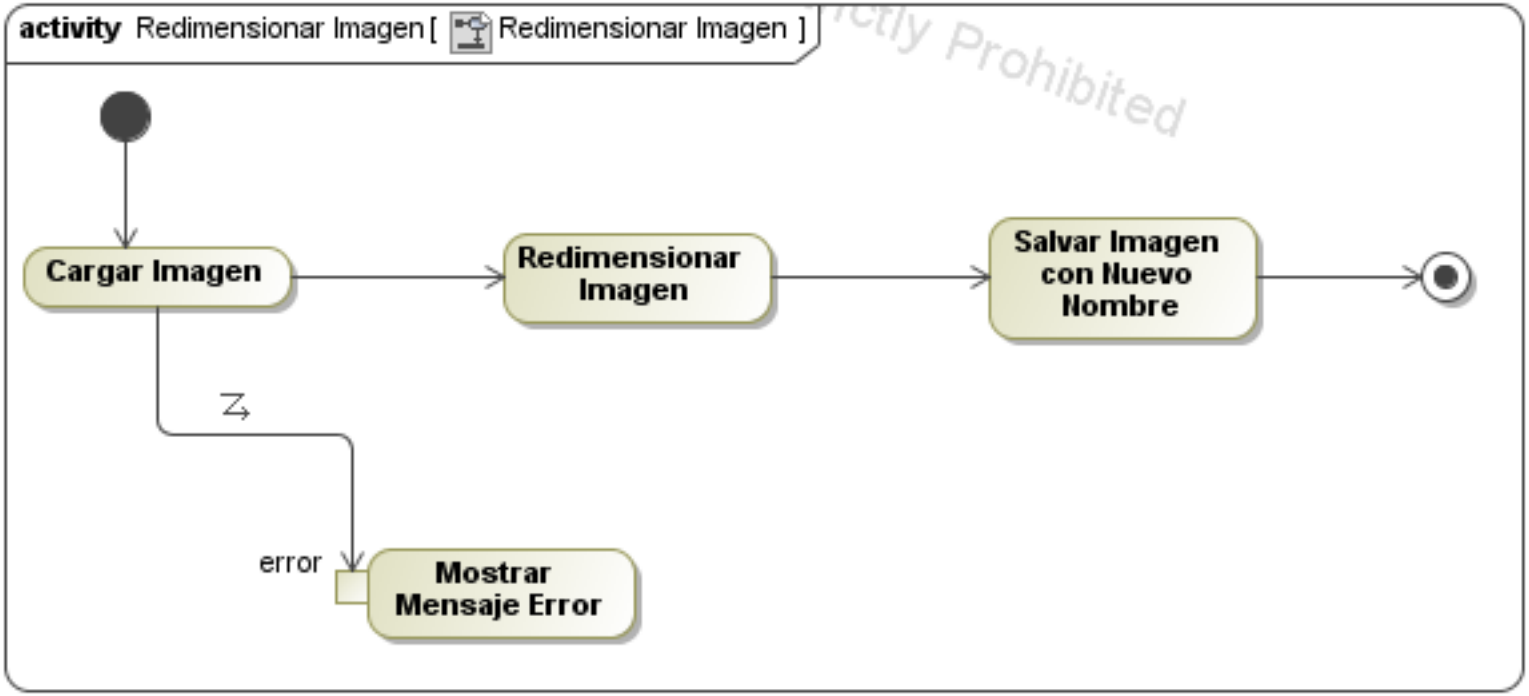
Sincronización

- Nodos de unión/división modelan actividades que pueden ocurrir simultáneamente o en cualquier orden



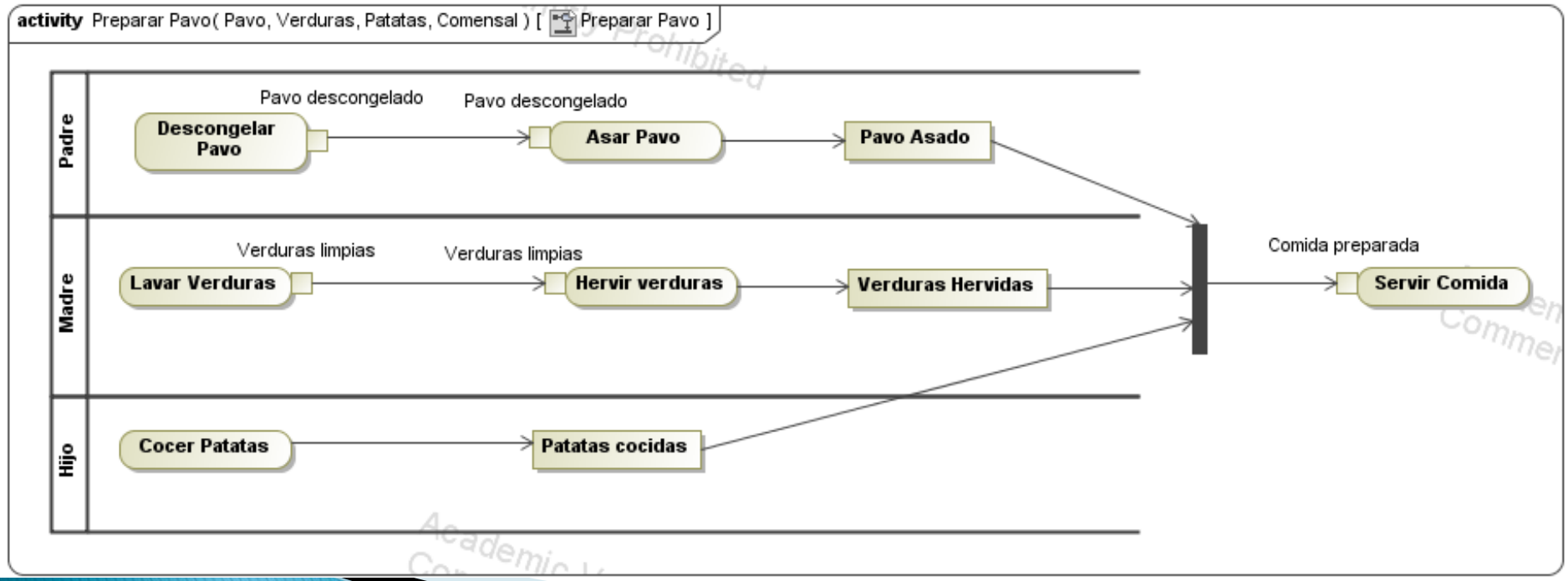
Excepciones

- Representan la ocurrencia de un error durante la ejecución de una actividad



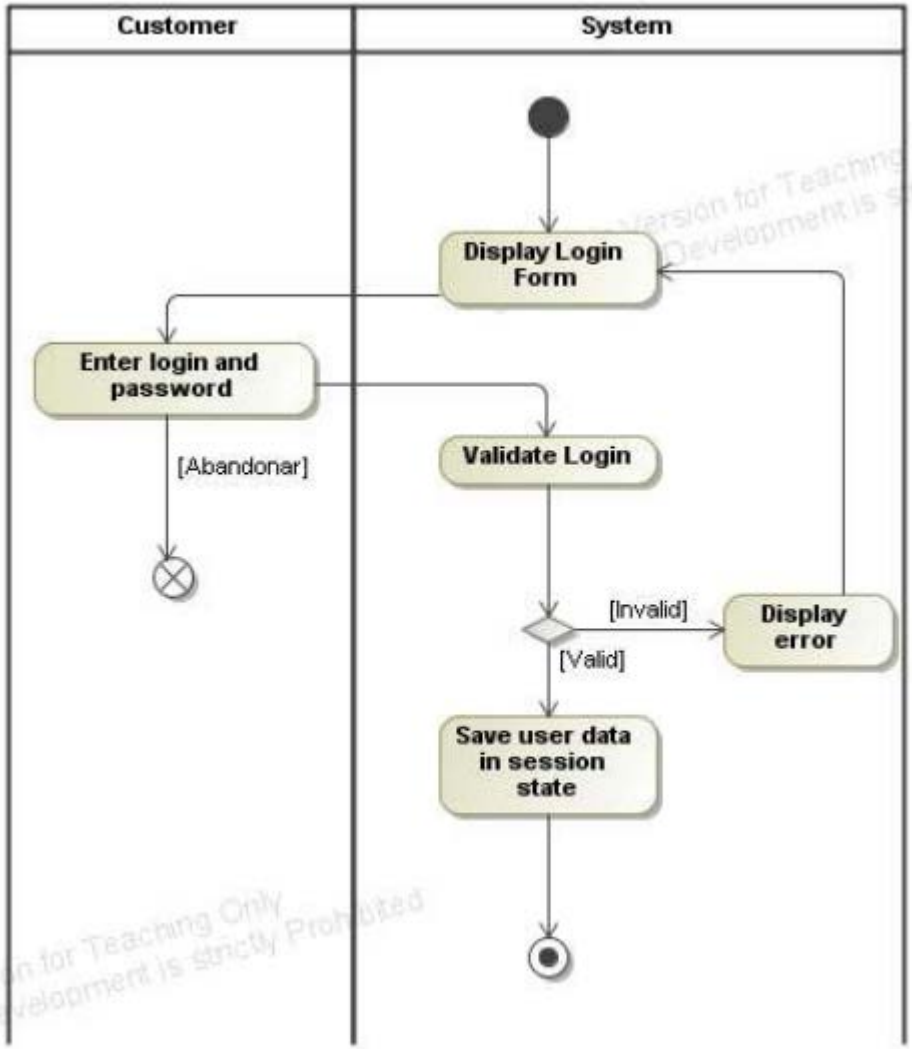
Swimlanes

- Las calles permiten agrupar las actividades según un criterio
- Asignan la responsabilidad de un comportamiento a usuarios, sistemas, objetos o capas dentro de una arquitectura

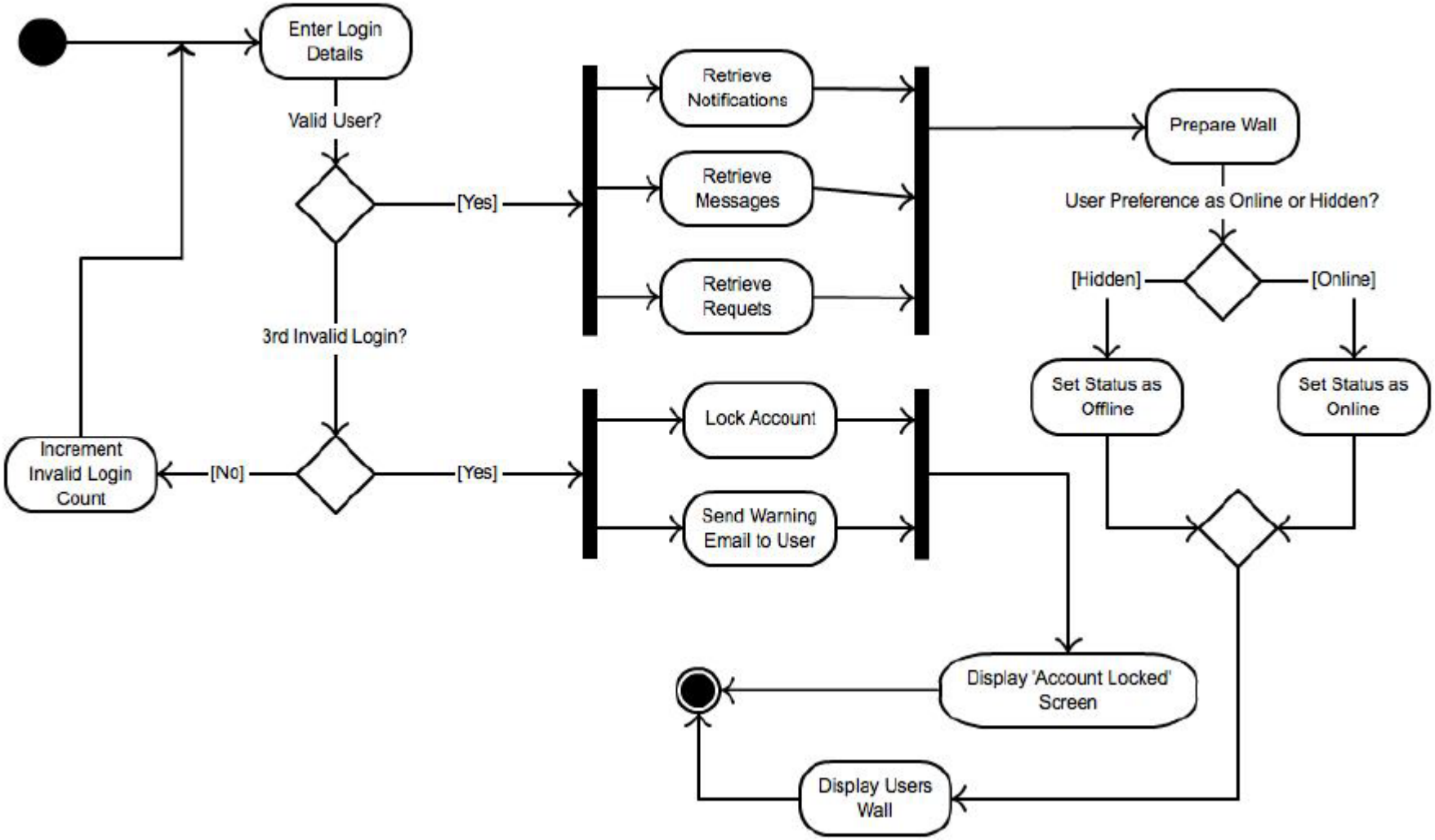


Swimlanes

- Ejemplo:



Ejemplo: Login de Facebook



Ejemplo: cadena de montaje de martillos

- En una planta de montaje se generan simultáneamente cabezas y mangos de martillos tras encender la corriente de la fábrica
  - Cuando se tiene una cabeza y un mango, se ensamblan ambas partes para generar un martillo
  - El martillo es luego pulido y termina el flujo

Ejemplo: Pedidos online

- En una aplicación online para realizar pedidos la actividad se inicia cuando se recibe una petición de un pedido desde el exterior
  - Una vez recibida la orden, el sistema confirma la recepción de dicha orden
  - Hay dos tipos de pedidos: normal y especial
    - El pedido puede ser de uno de estos dos tipos o de otro diferente
      - Si el pedido es normal, se confirma, se envía y se finaliza la actividad.
      - Si no es normal:
        - Si el pedido es especial, se confirma, se envía y se finaliza la actividad
        - Si el pedido no es especial, se cancela el pedido

Ejercicio: Biblioteca universitaria

- Vamos a modelar el préstamo y devolución de artículos en una misma actividad
- En nuestra biblioteca se pueden solicitar préstamos de artículos
  - Si se está fuera del periodo de préstamos, éste se deniega
  - Si no, se presta el artículo
    - Luego, hay que devolver el artículo
      - Si está dentro del plazo de devolución, no pasa nada, se coloca el artículo en su sitio
      - Si está fuera de plazo, se sanciona al prestatario
        - Si la suma de sus sanciones es mayor que 500, se suspende al prestatario y se coloca el artículo en su sitio
        - Si la suma es menor que 500, no se suspende al prestatario y se coloca el artículo en su sitio

Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

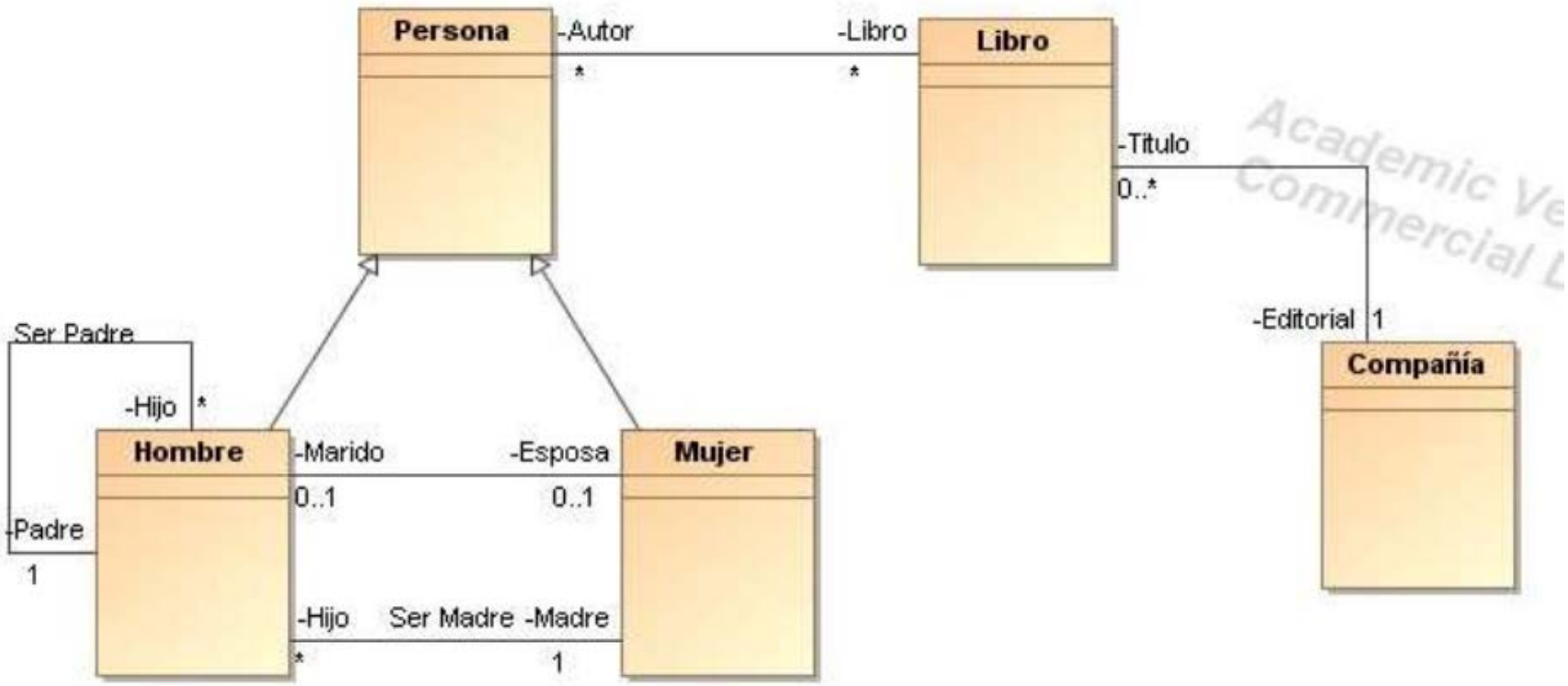
UML : Diagramas de Clase (Diagrama de Estructura)

Vistazo rápido

- Clases
- Objetos
- Atributos
- Operaciones
- Tipos Enumerados
- Clases abstractas
- Relaciones
- Ejemplos

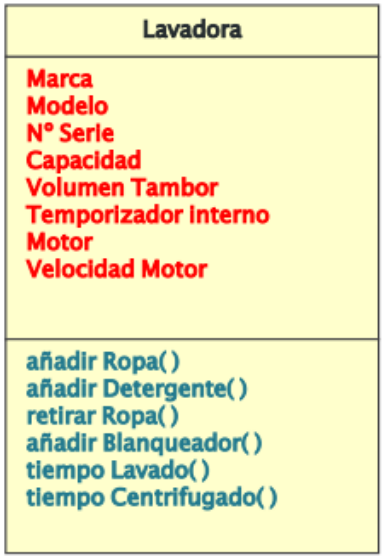
Clases

- Modelo de tipos de objetos y los atributos y relaciones permitidas



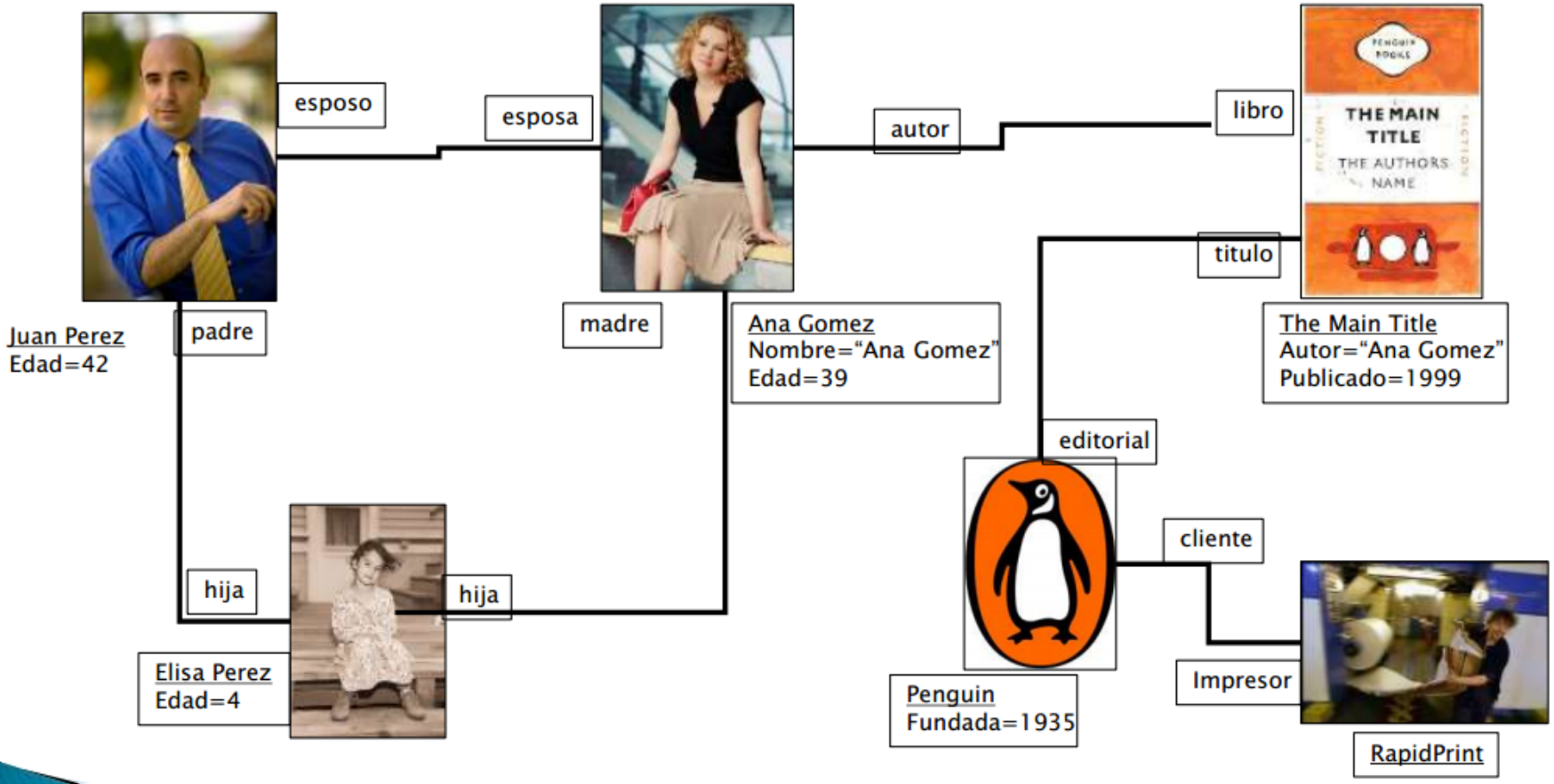
Clases

- Describe la estructura y el comportamiento de objetos que tienen las mismas características y semánticas
- La estructura se describe mediante sus **atributos**
- El comportamiento mediante sus **operaciones**



Objetos

Modelo de objetos: Valores de atributos reales y relaciones entre ellos

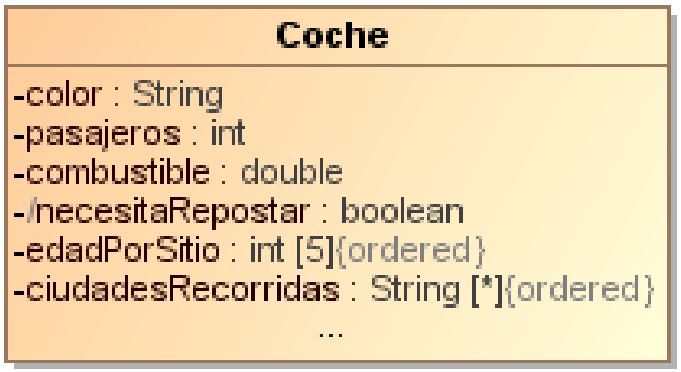


Objetos

- Un objeto es una **instancia** de una clase
- Un objetos puede ser cualquier elemento útil con identidad, estado y funcionamiento propios
- En UML se permiten objetos anónimos

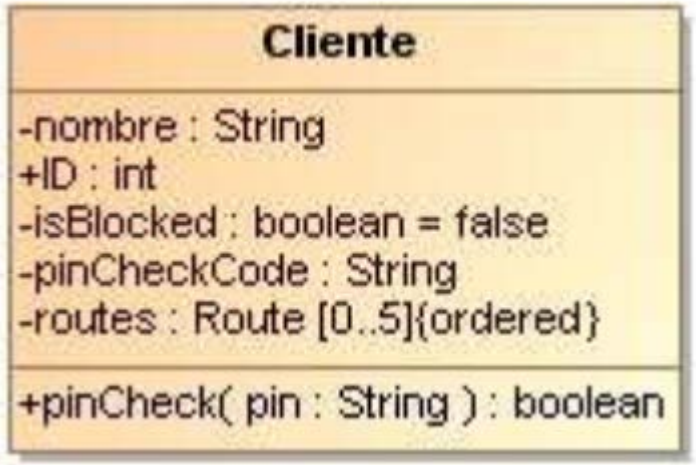
Atributos

- Representan la información almacenada en una clase
- Tipos simples o relaciones con otros objetos
- Representados como atributos integrados o por relación
- Su nombre empieza con minúscula



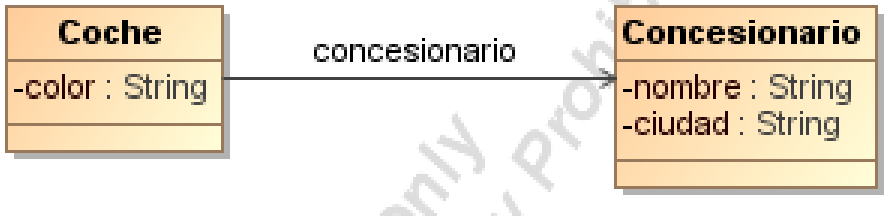
Atributos

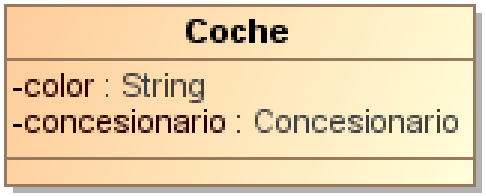
- Un atributo define una propiedad estructural de una clase
- La descripción se compone de:
  - Visibilidad
  - Nombre
  - Tipo
  - Multiplicidad



Atributos

- Ejemplo de un atributo por relación

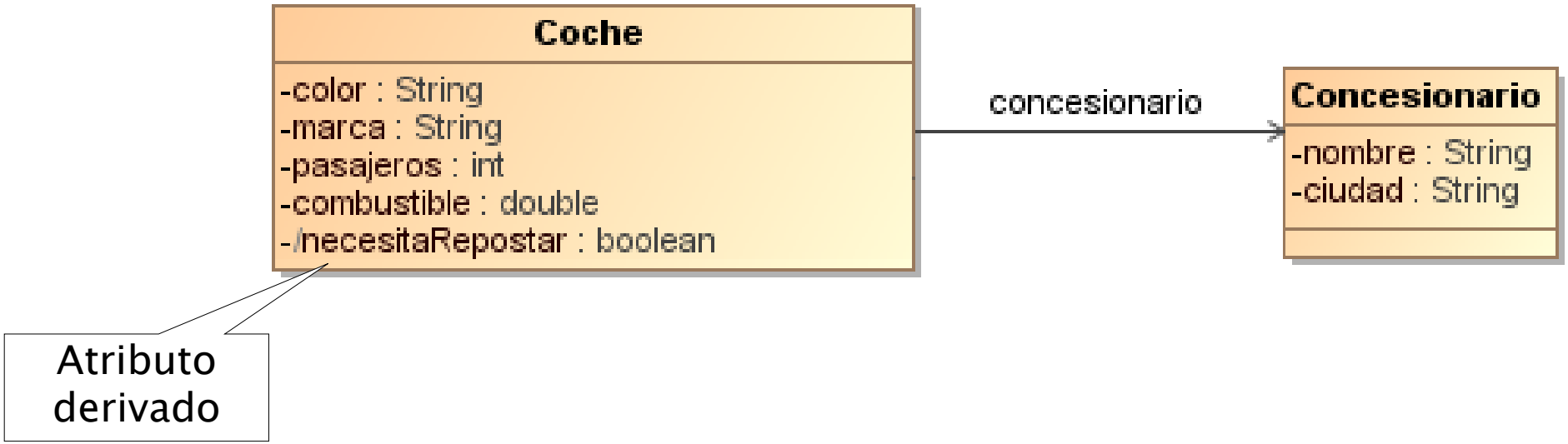




Vamos a preferir esta representación

Atributos

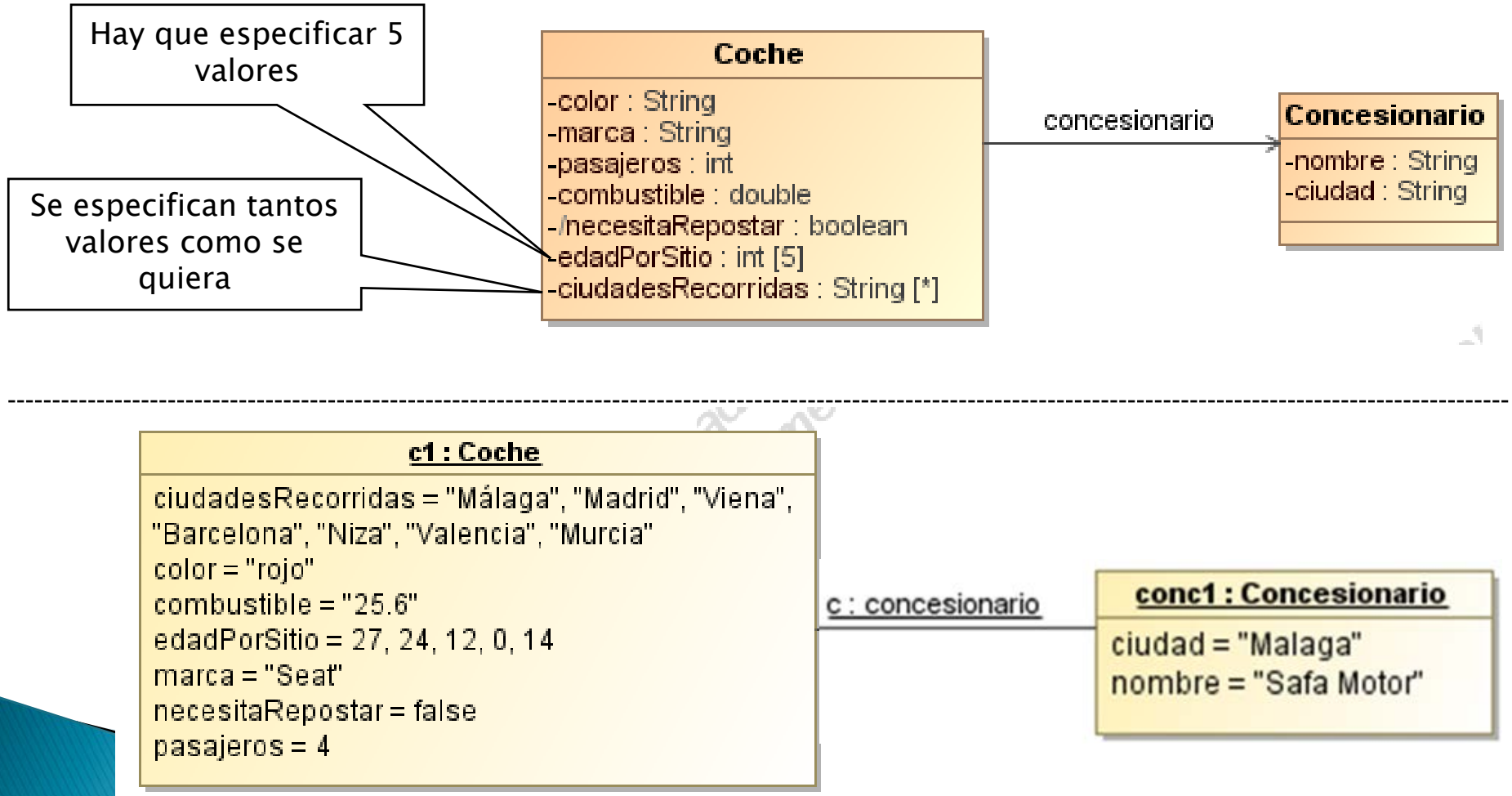
- Atributos derivados
  - Son aquellos cuyo valor se puede calcular a partir del de otros atributos.



  - Se puede saber si un coche necesita repostar de acuerdo a la cantidad de combustible que tenga

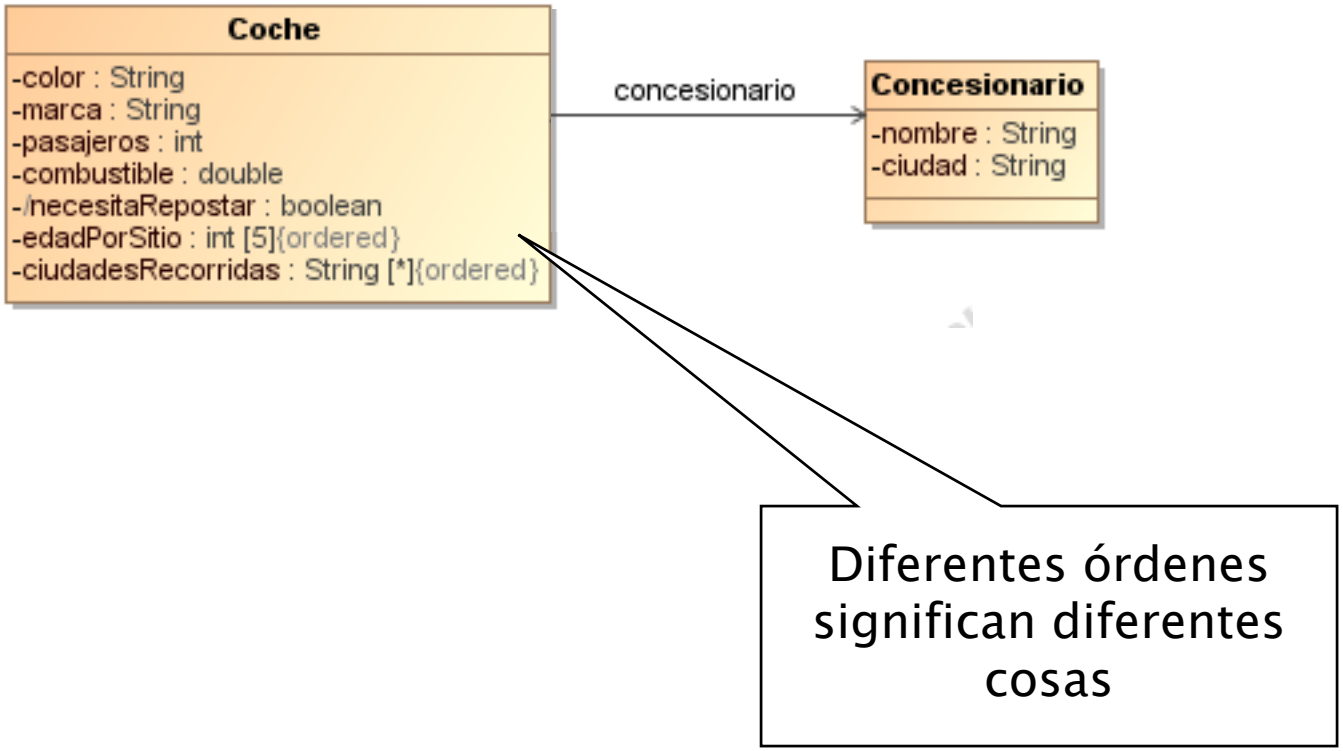
Atributos

- Multiplicidad en los atributos



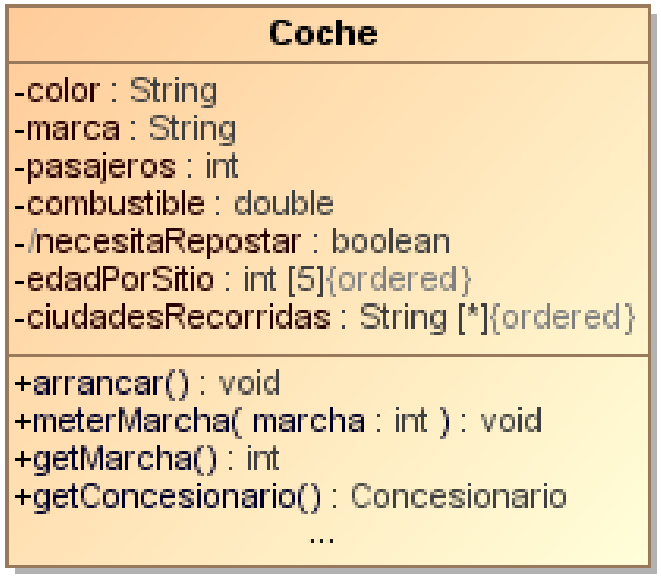
Atributos

- Ejemplo de atributo como conjunto ordenado



Operaciones

- Especifican las acciones que las instancias de una clase pueden realizar
  - Su nombre comienza con minúscula
  - Pueden devolver algo (*función*), o simplemente realizar cambios en el objeto (*procedimiento*)



Operaciones

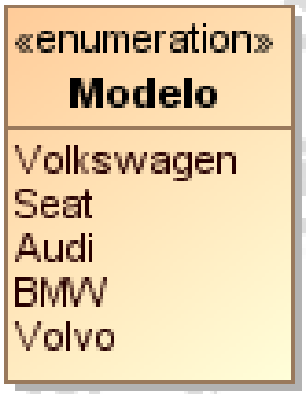
- Las operaciones también pueden especificar restricciones
  - Características que la implementación de la operación debe cumplir (contrato)
  - Precondiciones y postcondiciones
  - Se pueden describir junto a la operación (entre {}) o mediante una nota
  - Operaciones de consulta *(query operation)*

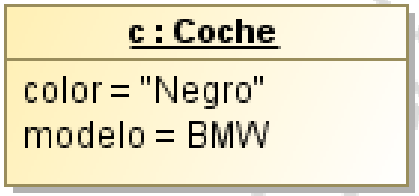
Tipos

- Hemos visto que los tipos de los atributos pueden ser:
  - Tipos simples: boolean, int, String …
  - Otras clases definidas en nuestro diagrama de clases

- También existen los **tipos enumerados**
  - Los atributos de dichos tipos sólo pueden adquirir los valores determinados en el tipo enumerado

Tipos enumerados

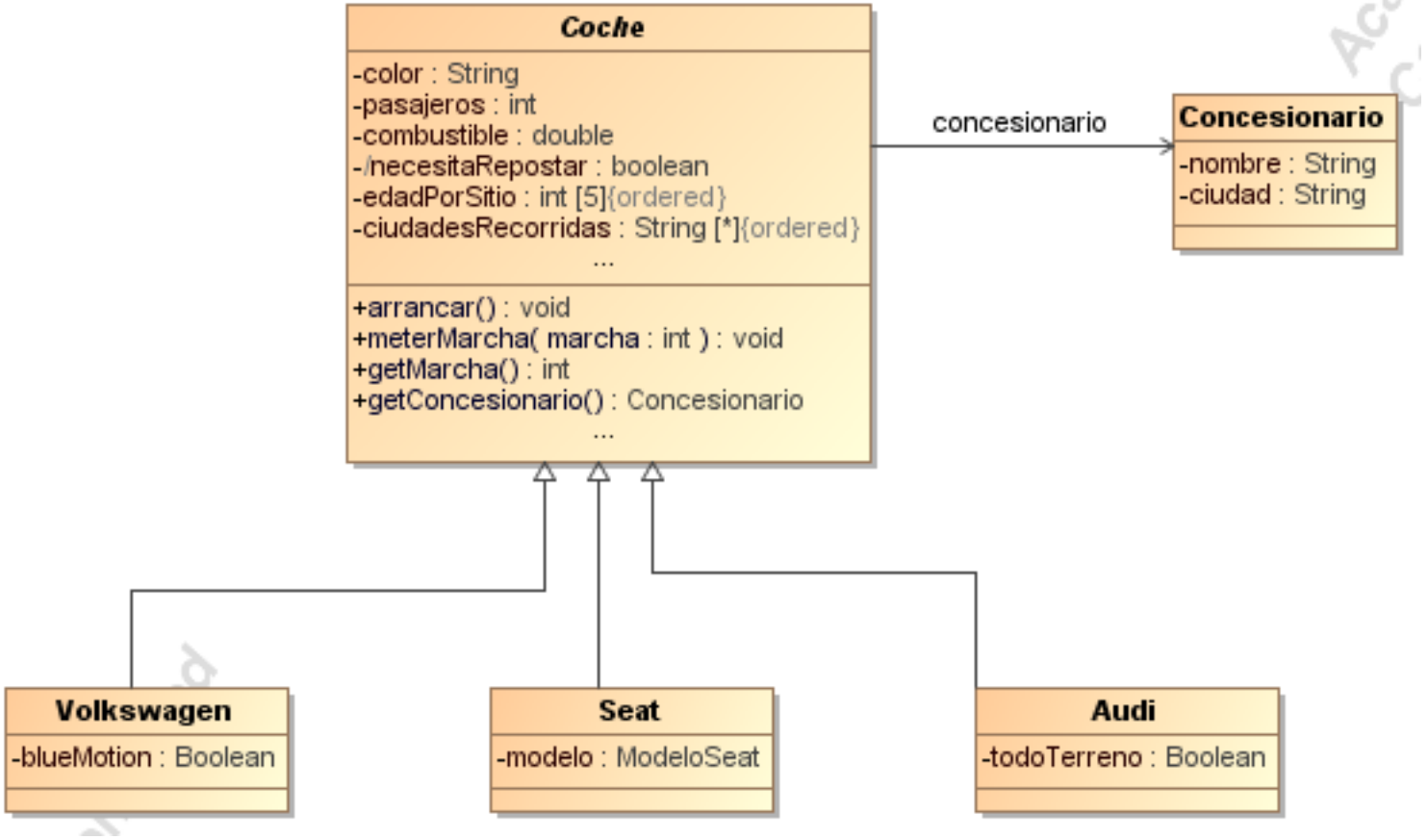




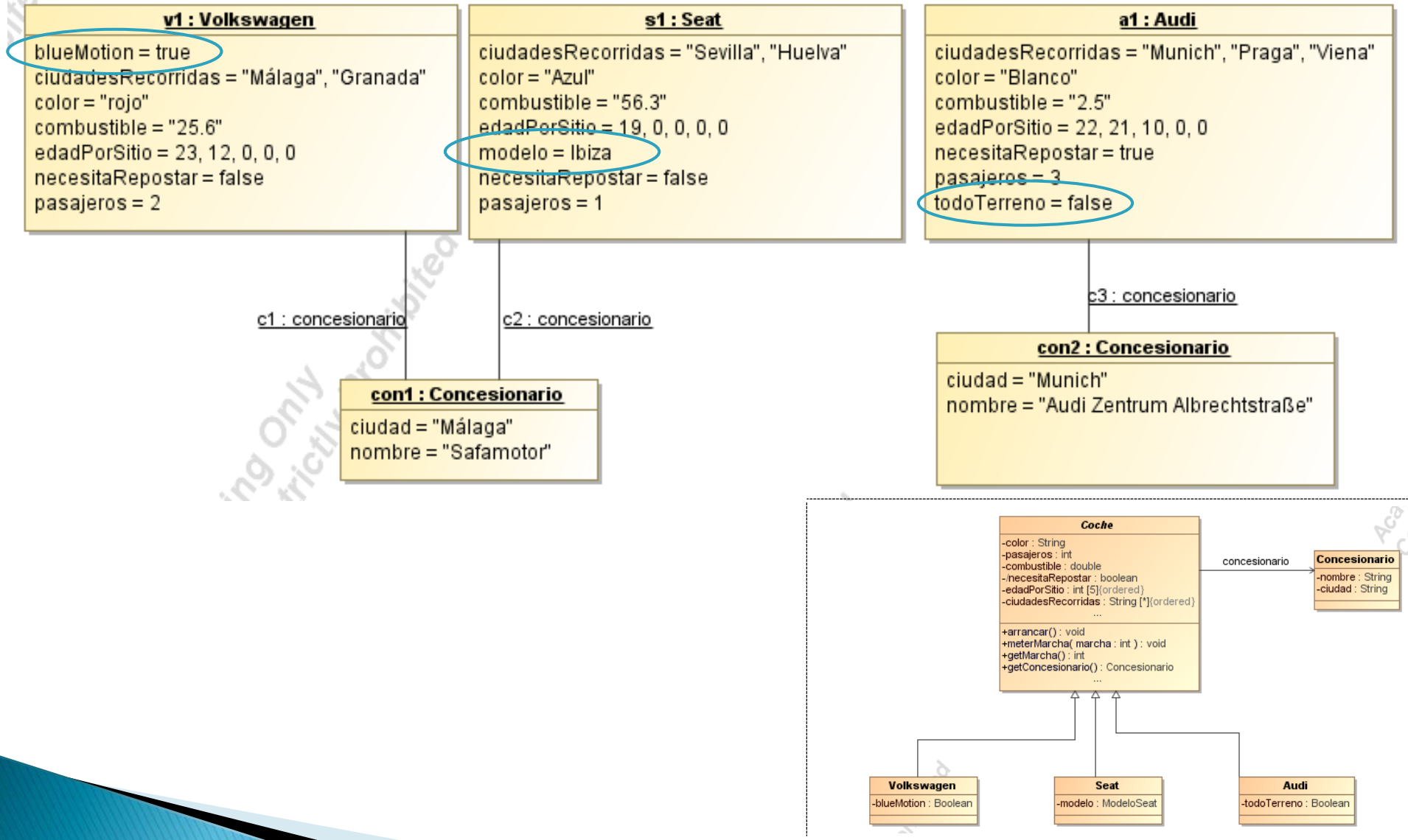
Clases abstractas: herencia

- Las clases abstractas se utilizan como súperclases de otras clases no abstractas
- Se representan con letra cursiva
- Las subclases adquieren **todos** los atributos y operaciones de la súperclase
  - Además, pueden redefinir nuevos: **especialización**
- No puede haber instancias de las clases abstractas

Clases abstractas: herencia



Clases abstractas: herencia



Relaciones

- Son los mecanismos básicos de comunicación entre clases
- Tipos:
  - Asociación
  - Agregación
  - Composición
  - Especialización

Relaciones

- **De asociación**
  - Relación de uso persistente
  - Se puede leer como “un objeto está asociado a otro”

  - En nuestro caso: “un coche está asociado a un concesionario”

Relaciones

- Asociación: Relación estructural entre dos clases
- Permite a los objetos de una clase contactar los objetos de otra clase para acceder a sus datos y su comportamiento

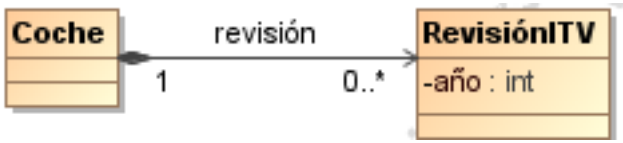
- Bidireccionales o unidireccionales

- Se caracterizan por:
  - Nombre
  - Roles
  - Multiplicidad (ej.: 0..1, 1, 2..4, 0..*, 1..*)

Relaciones

Composición

- Relación fuerte
- Se representa con un rombo con relleno
- Modela la relación entre un todo y las partes
- Se puede leer como "… es parte de …"
- El tiempo de vida del objeto incluido está condicionado por el tiempo del vida del que lo incluye
- El objeto incluido no tiene sentido sin el que lo incluye

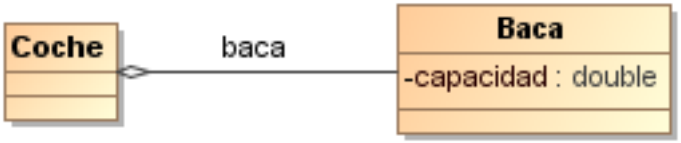


- En el ejemplo, los objetos de tipo RevisiónITV son parte de un objeto de tipo Coche
  - No puede haber RevisiónITV si no hay Coche

Relaciones

Agregación

- Relación más débil que la anterior
- Se representa con un rombo sin relleno
- Se puede leer como "… posee un … "
- El tiempo de vida del objeto incluido no está condicionado por el tiempo del vida del que lo incluye
- El objeto incluido sí tiene sentido sin el que lo incluye



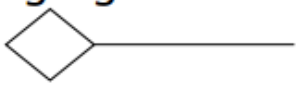
- En el ejemplo, los objetos de tipo Baca están relacionados con un objeto de tipo Coche por una relación de agregación
  - Si el Coche desaparece, la Baca no desaparece
  - La Baca tiene sentido, como objeto, sin el Coche

Relaciones

- Tres tipos de relación

Asociación "simple"

Agregación

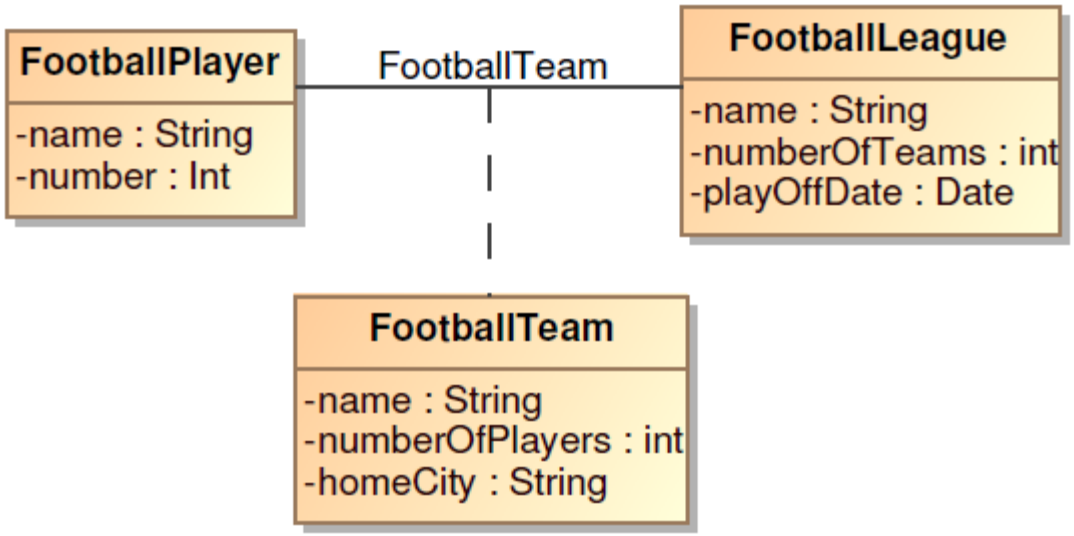


Composición

- Asociación
  - Los objetos son conscientes unos de otros
- Agregación
  - Punto de control
  - Actúa como una unidad
- Composición
  - Una parte sólo puede ser miembro de una agregación

Clases de asociación

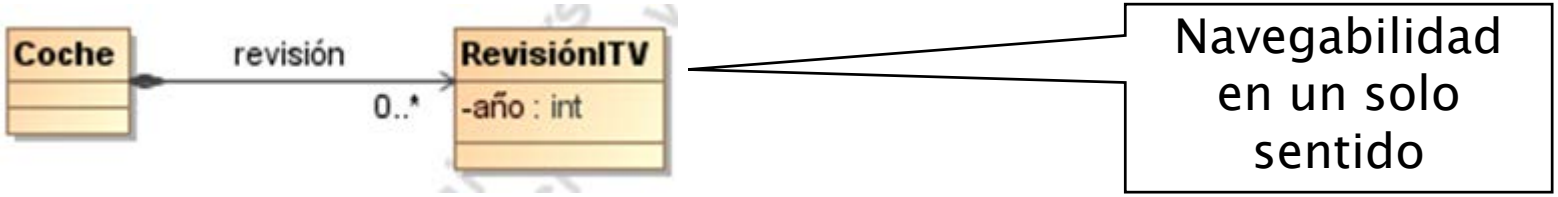
- Sirven para modelar relaciones con características complejas
- Suelen codificarse como tres clases
  - Dos de ellas son las clases a relacionar
  - La tercera describe la relación entre las dos anteriores

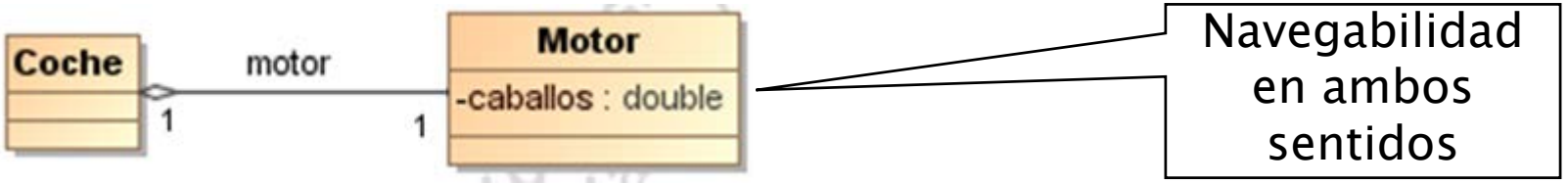


Multiplicidad de atributos y relaciones

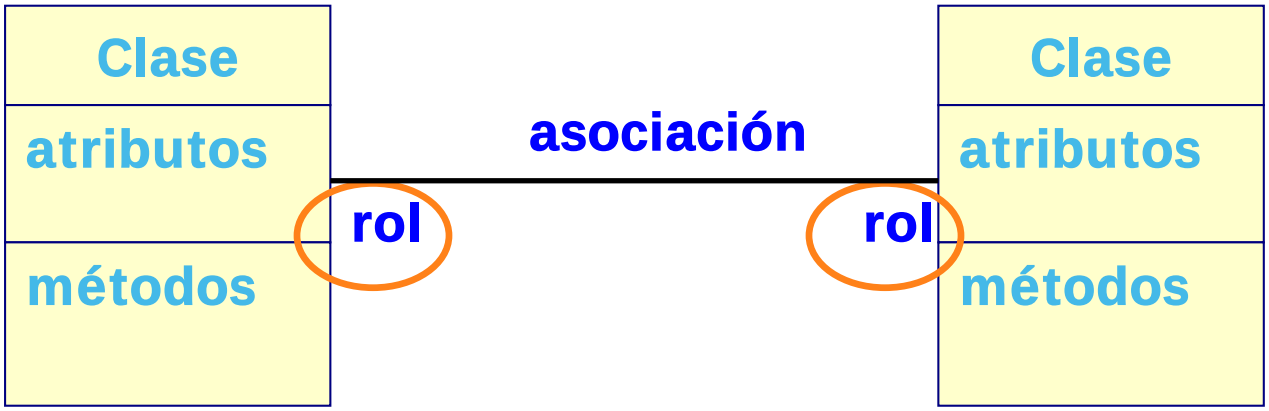
Multiplicidad	Significado
0..1	El valor es opcional
1	Exactamente uno (por defecto)
2..4	Al menos 2 y como máximo 4
0..*	Numero arbitrario de valores
*	Otra forma del anterior

- **Navegabilidad en las relaciones**
  - Se refiere a la posibilidad de acceso de un extremo a otro de la relación. Se indica con flechas

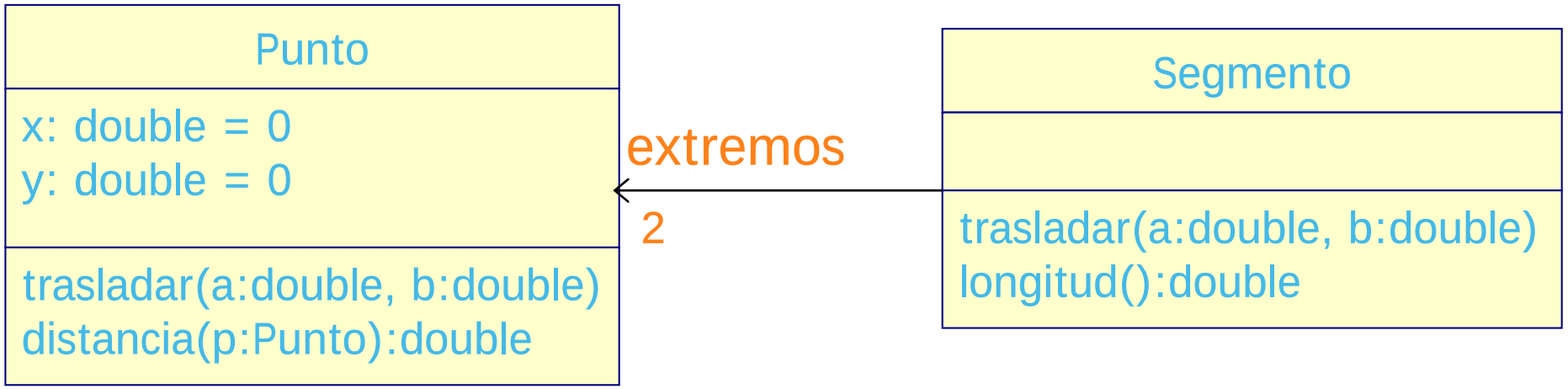




Roles en relaciones



Roles en relaciones

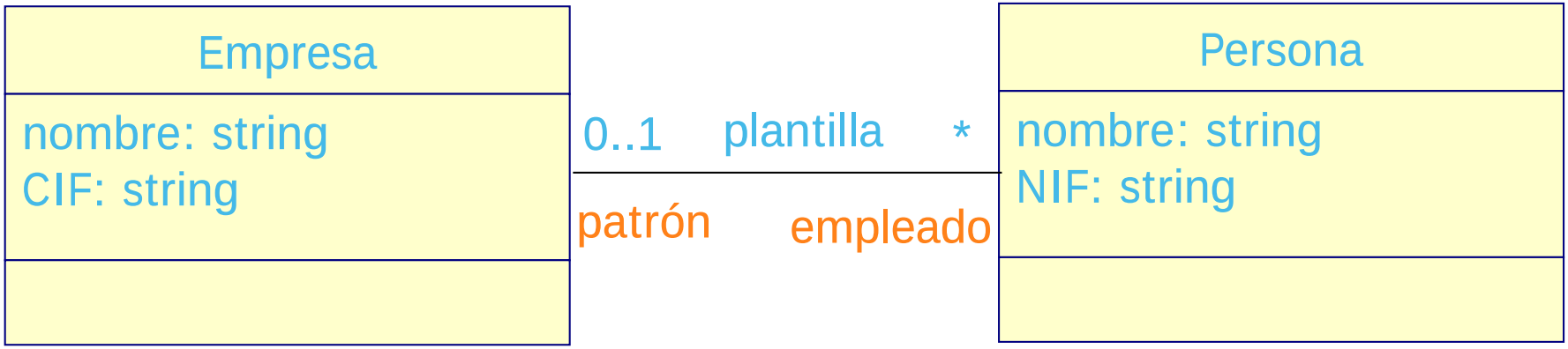


Tenemos un objeto de tipo **Segmento** cuyo identificador es **seg**

Al escribir **seg.extremos**, obtenemos una colección con dos objetos de tipo **Punto**

Si no se pone nombre en el rol, entonces éste es el nombre de la clase (comenzando por minúscula): **seg.punto**

Roles en relaciones

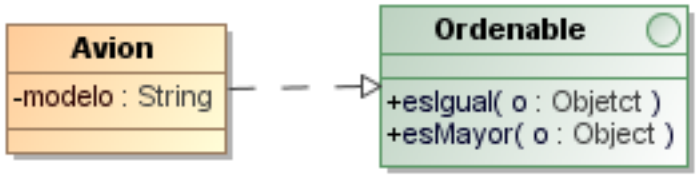


Si tenemos un objeto de tipo **Empresa** con identificador **emp**, al escribir **emp.empleado** obtenemos una colección, que puede ser vacía, de **Personas**.

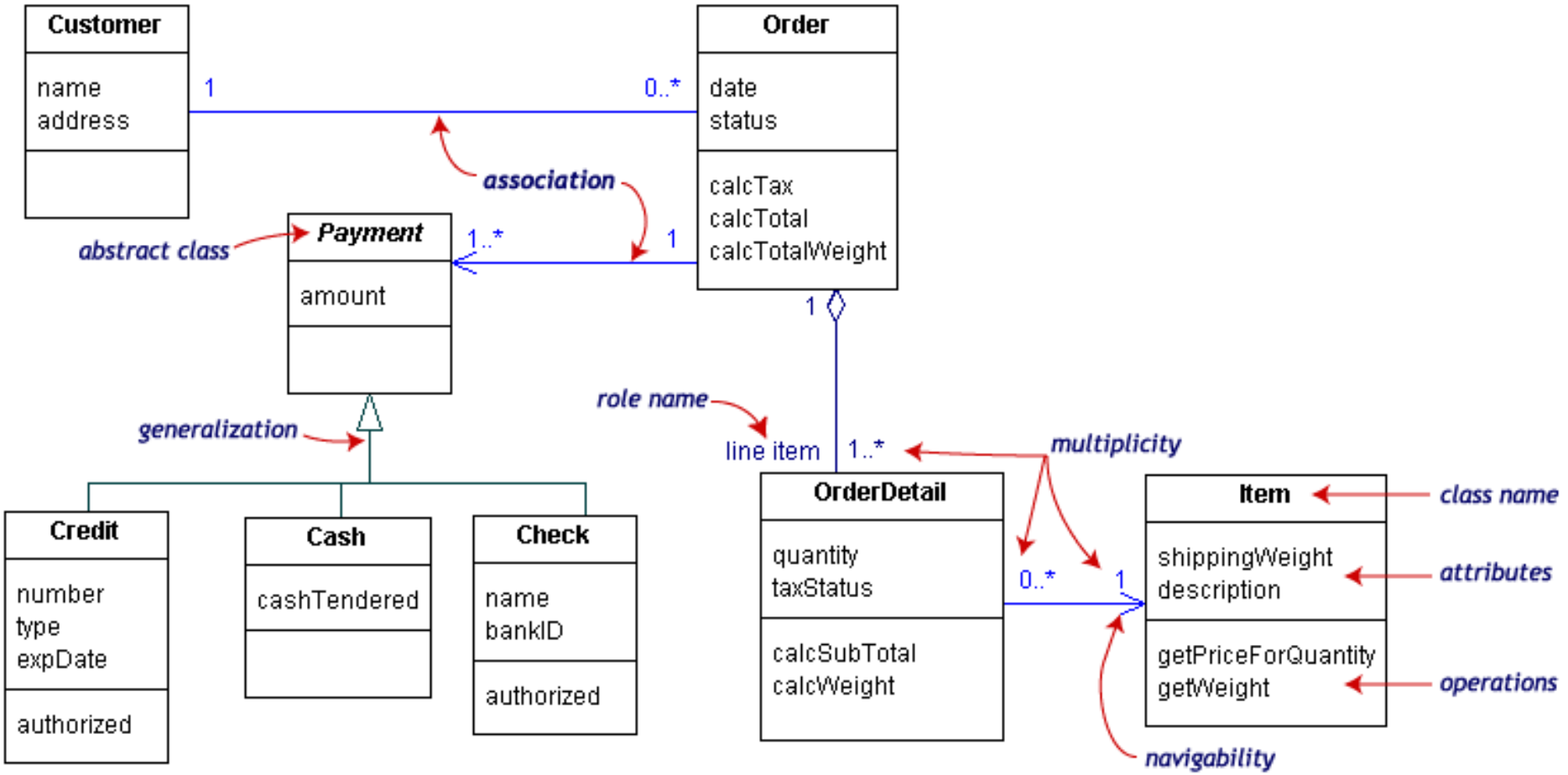
Si tenemos un objeto de tipo **Persona** con identificador **p**, al escribir **p.patrón** obtendremos, o no, un objeto de tipo **Empresa**

Interfaces

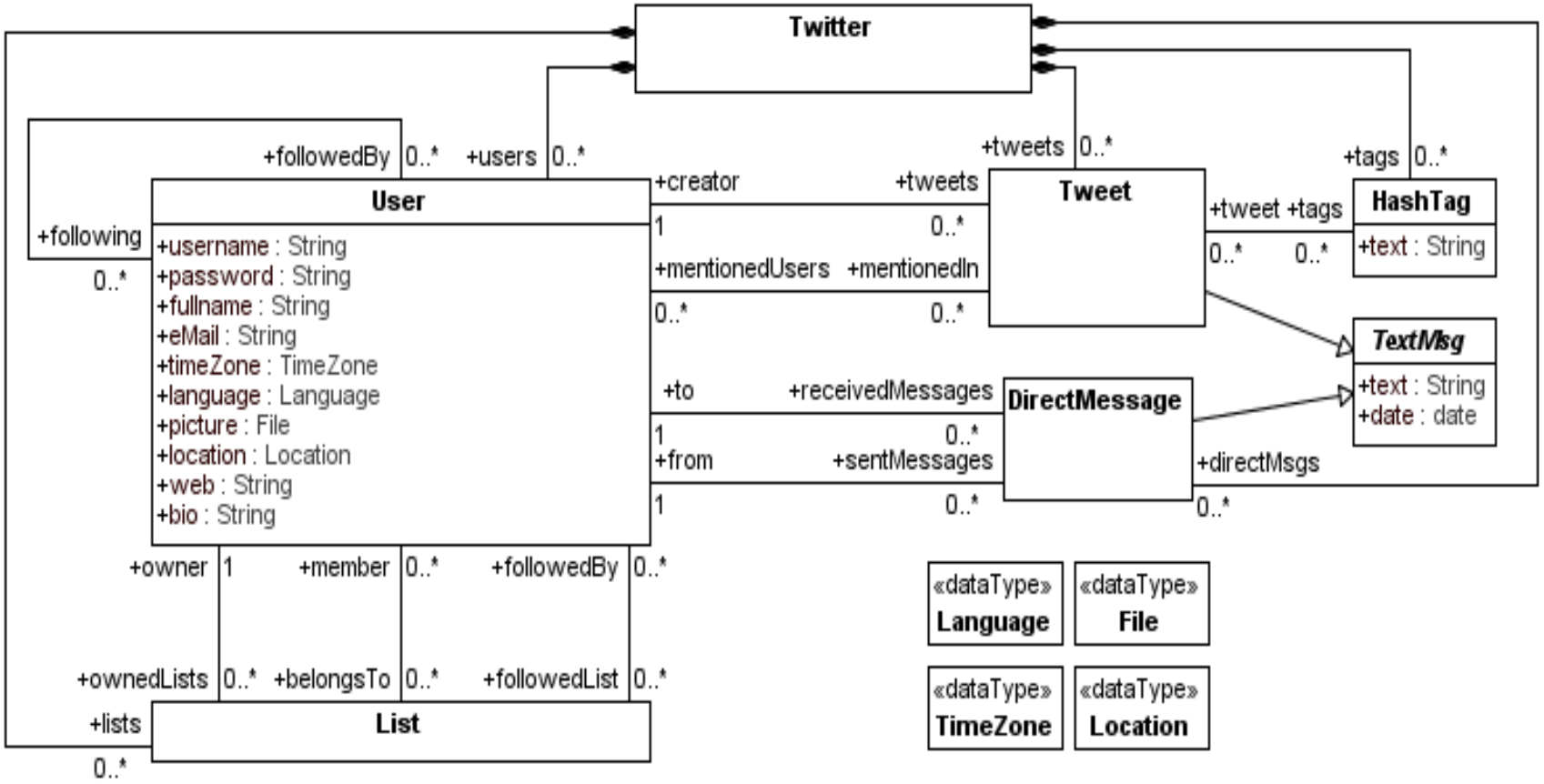
- Una interfaz es una colección de operaciones que se usa para especificar un servicio de una clase o componente
- Se puede ver como equivalente a una clase abstracta
- Las operaciones no se definen en la interfaz, sino en cada una de las clases que implementan la interfaz



Ejemplo


Customer
name
address
1
0..*
Order
date
status
calcTax
calcTotal
calcTotalWeight
association
abstract class
Payment
amount
1..*
1
1
role name
line item
1..*
multiplicity
generalization
Credit
number
type
expDate
authorized
Cash
cashTendered
Check
name
bankID
authorized
OrderDetail
quantity
taxStatus
calcSubTotal
calcWeight
0..*
1
Item
shippingWeight
description
getPriceForQuantity
getWeight
class name
attributes
operations
navigability

Ejemplo: Parte del metamodelo de Twitter



Ejemplo: Cadena de montaje de matillos (I)

- Una cadena de montaje se va a componer de elementos (máquinas y bandejas) que tienen una posición concreta dentro de la misma (coordenadas x,y)
  - Las máquinas tendrán un tiempo de procesado determinado y, a su vez, son de tres tipos:
    - Generadores de piezas. Tienen un contador y, a su vez, son de dos tipos
      - Generadores de mangos
      - Generadores de cabezas
    - Ensambladores de martillos
    - Pulidoras de martillos

Ejemplo: Cadena de montaje de matillos (II)

- Las bandejas tienen una capacidad determinada y pueden contener piezas con un orden determinado
  - Las piezas son de tres tipos:
    - Cabezales de martillo
    - Mangos de martillo
    - Martillos, los cuales pueden estar pulidos o no.

Relaciones entre máquinas y bandejas:

- Las máquinas pueden tener o no bandejas como entrada y siempre van a tener una bandeja como salida
- Las bandejas pueden desembocar o no en máquinas y siempre tendrán una o más máquinas como entrada

Ejemplo: Red (I)

- Tenemos una red compuesta por componentes y paquetes. Los primeros constan de un identificador y pueden procesar un paquete a la vez, y de los segundos se tiene información de si están siendo procesados y tienen también un sello de tiempo
  - Los componentes pueden tener otros componentes por vecinos y, a su vez, se componen de paquetes
  - Hay tres tipos de componentes:
    - Usuarios, que llevan la cuenta del nº de paquetes enviados
    - Servidores, que llevan la cuenta del nº de paquetes recibidos
    - Nodos (inf en siguiente transparencia)

Ejemplo: Red (II)

- Los nodos llevan la cuenta del número de paquetes procesados
  - Hay un tipo especial de nodos que son los nodos de apoyo cuya misión es activarse cuando la red está sobrecargada. Estos nodos sólo están activos a veces, por lo que tienen información de si están activos o no y también llevan la cuenta de los paquetes que han procesado. Además, tienen información de los nodos que apoyan.

Ejercicio : Biblioteca universitaria (I)

- La biblioteca se compone de un bibliotecario (un bibliotecario sólo trabaja en una biblioteca) y de prestatarios (pueden ser prestatarios de varias bibliotecas) que pueden pedir prestados artículos. Los prestatarios pueden ser profesores, estudiantes o estudiantes de postgrado. Los artículos, que sólo están en una biblioteca, pueden ser libros o revistas.
- La biblioteca tiene periodos de préstamos y tiene usuarios suspendidos por no haber respetado los plazos de entrega

Ejercicio : Biblioteca universitaria (II)

- Datos sobre los atributos
  - La biblioteca tiene información sobre su dirección, periodos de préstamos, número máximo de préstamos y los usuarios suspendidos.
  - De los prestatarios se indica su nombre, dirección, número de artículos prestados, sanciones y su estado (que puede ser *suspendido* o *liberado*)
    - De un profesor se indica la facultad y departamento
    - De un estudiante de postgrado se indica su departamento y tutor
    - De un estudiante se indica su tutor y facultad
  - De los artículos se tiene información de su ubicación, estado (*prestado* o *disponible*) y título
    - Los libros indican además su autor, ISBN y año de publicación
    - Las revistas contienen información sobre su fecha de publicación, ISSN y número.

Ejercicio : Biblioteca universitaria (III)

- Datos sobre las operaciones
  - El bibliotecario puede prestar un artículo a un prestatario, devolverlo y cobrar sanciones de cierta cantidad a un prestatario
  - Los prestatarios pueden solicitar el préstamo de un artículo y devolver el artículo
- Datos sobre las relaciones
  - En un préstamo entre un prestatario y un artículo, se tiene información sobre la fecha en que se realiza el préstamo y la que se debe devolver.
  - Permitir navegabilidad en todas direcciones

Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

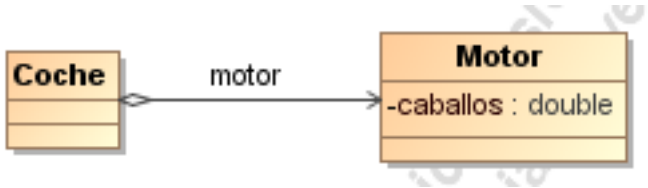
Nathalie Moreno
moreno@lcc.uma.es

UML : Diagramas de Estructuras Compuestas

(Diagrama de Estructura)

Introducción (I)

- Como se vio en los diagramas de clase, unos objetos pueden estar compuestos por otros
  - UML nos permite modelar esta información por medio de relaciones de composición entre objetos contenedores y sus partes
  - Se muestra con un diamante en el contenedor, en una relación entre el contenedor y la parte

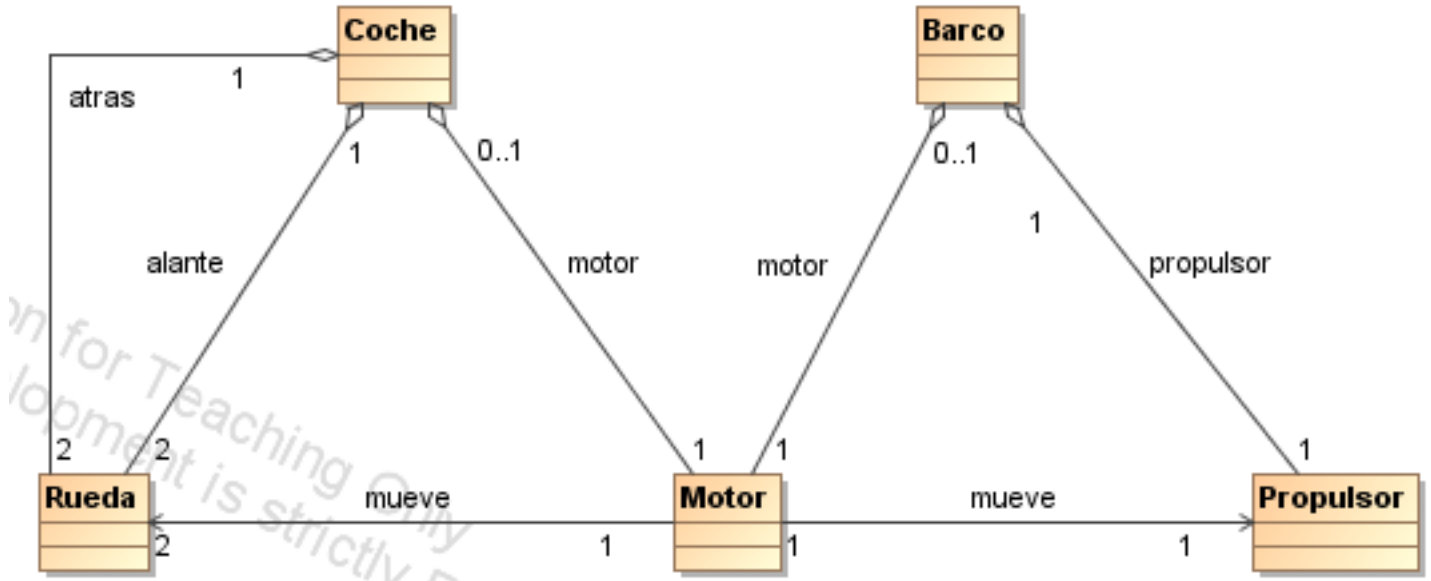


Introducción (II)

- Modelar composiciones en UML se vuelve complejo en ciertas situaciones
- Ejemplo
  - Tenemos coches y barcos ambos compuestos por un motor
    - El motor del coche mueve las ruedas delanteras del vehículo
    - El motor del barco mueve el propulsor del barco

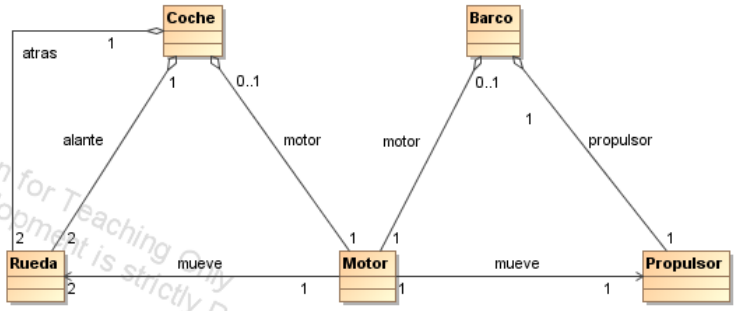
Introducción (III)

- Habría que realizar un modelo complejo para aclarar (a quien pudiera leer el diagrama), que hay una diferencia entre el motor del coche y el del barco.



Introducción (IV)

- El diagrama de clases mostrado tiene deficiencias:
  - Con la relación de multiplicidad entre Coche y Motor y Barco y Motor se modela que el Motor sólo puede estar en uno de los dos, aunque no se consigue del todo. Además:
    - Parece que una instancia del motor de un coche puede mover tanto un propulsor como las ruedas y lo mismo con el del barco
    - Tampoco aclara que las dos ruedas que mueve en el caso del coche son las traseras y no las delanteras



Introducción (V)

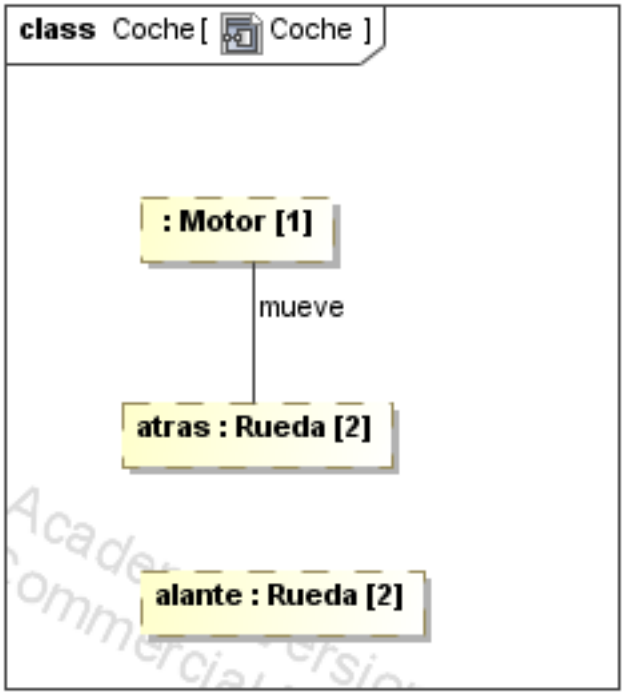
- Para modelarlo correctamente en un diagrama de clases tendríamos que elaborar toda una jerarquía de herencia entre clases para distinguir entre los motores de barcos y coches, y entre las ruedas delanteras y traseras de un coche, o marcando dependencias entre las relaciones.

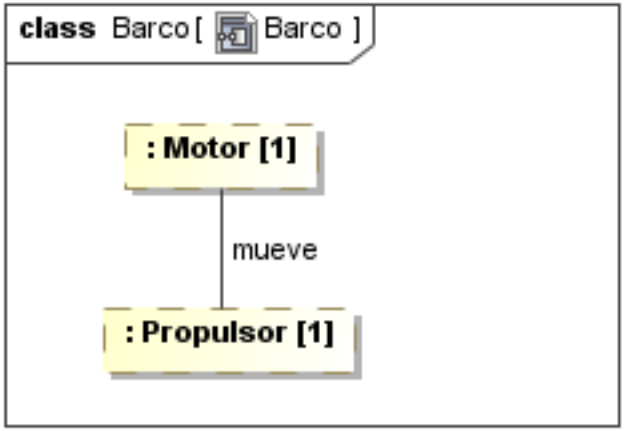
Solución: diagramas de estructuras compuestas (I)

- Permiten contextualizar las partes que componen una clase
- Se puede definir un diagrama donde se aclara que el coche tiene un motor que mueve las dos ruedas traseras
- Se tendrá otro diagrama donde se especifica que el barco tiene un motor que exclusivamente mueve su propulsor

Solución: diagramas de estructuras compuestas (II)

- El contexto lo define la clase contenedora
  - En este caso son el coche o el barco
- Dentro de la clase contenedora aparecen los objetos que la componen





Estructura compuesta

- Estructura. Conjunto de elementos conectados que proporcionan una funcionalidad colectiva.

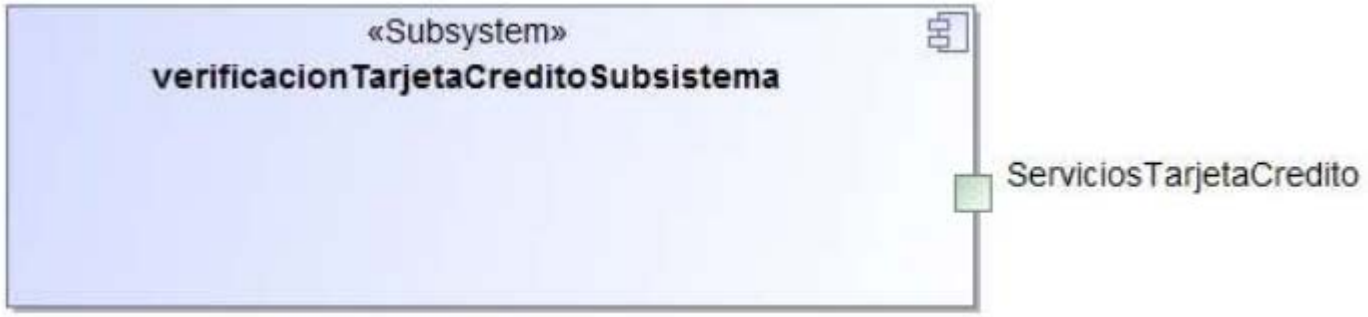
- Sirven para representar el funcionamiento interno de elementos complejos, como subsistemas o componentes

Elementos que componen el diagrama

- Clase: elemento del cual se ilustra su composición interna. También se les llama *clasificadores estructurados*
- Parte: objetos que conforman al elemento principal, se muestra con un rectángulo
- Conector: relación entre los elementos internos de la clase
- Puertos: indican la entrada o salida de una parte hacia otra parte y también con el mundo exterior

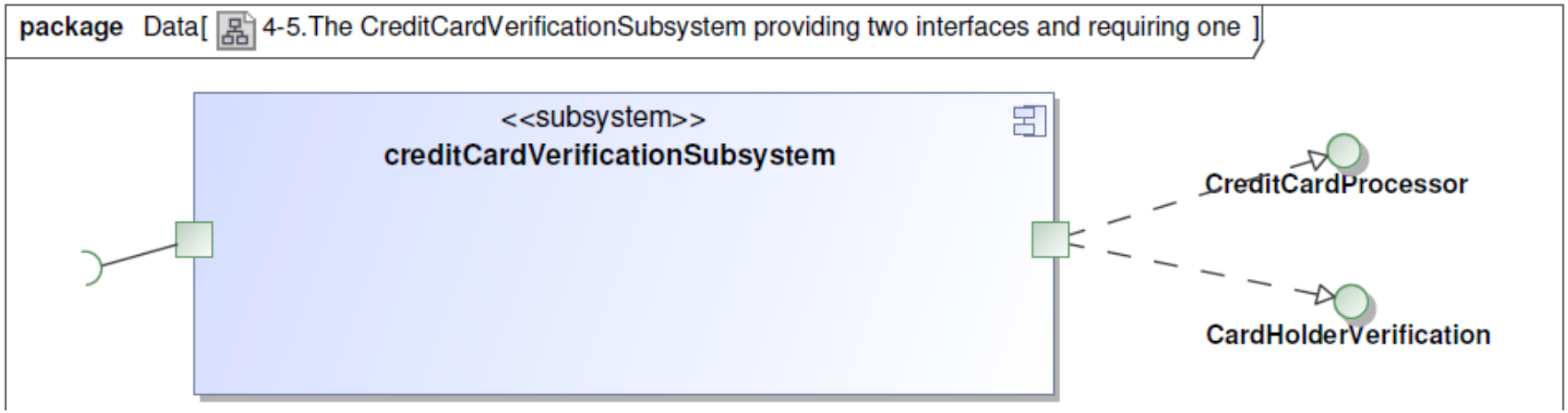
Puertos. Representación.

- Puerto: Mecanismo de comunicación de una estructura compuesta con el exterior
- Ofrece una de las funcionalidades de la estructura compuesta
- Oculta el mecanismo interno (abstrae) que lleva a cabo dicha funcionalidad
- Puede ser usado por otros elementos que respondan a la especificación del puerto



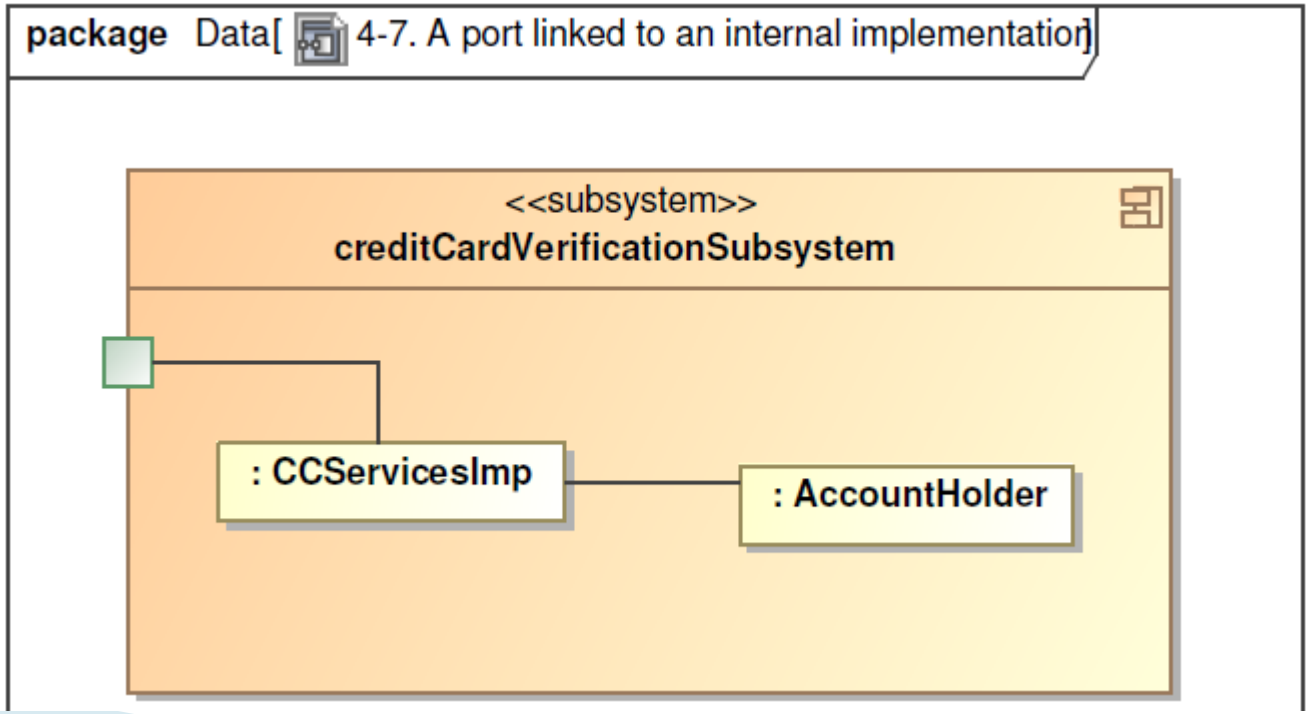
Puertos: comunicación con el exterior (I)

- Los puertos se asocian a interfaces (requeridos u ofrecidos)
  - La clase implementa las operaciones de las interfaces ofrecidas (representadas con círculo)
  - La clase necesita las operaciones de las interfaces requeridas (representadas con semicírculo)



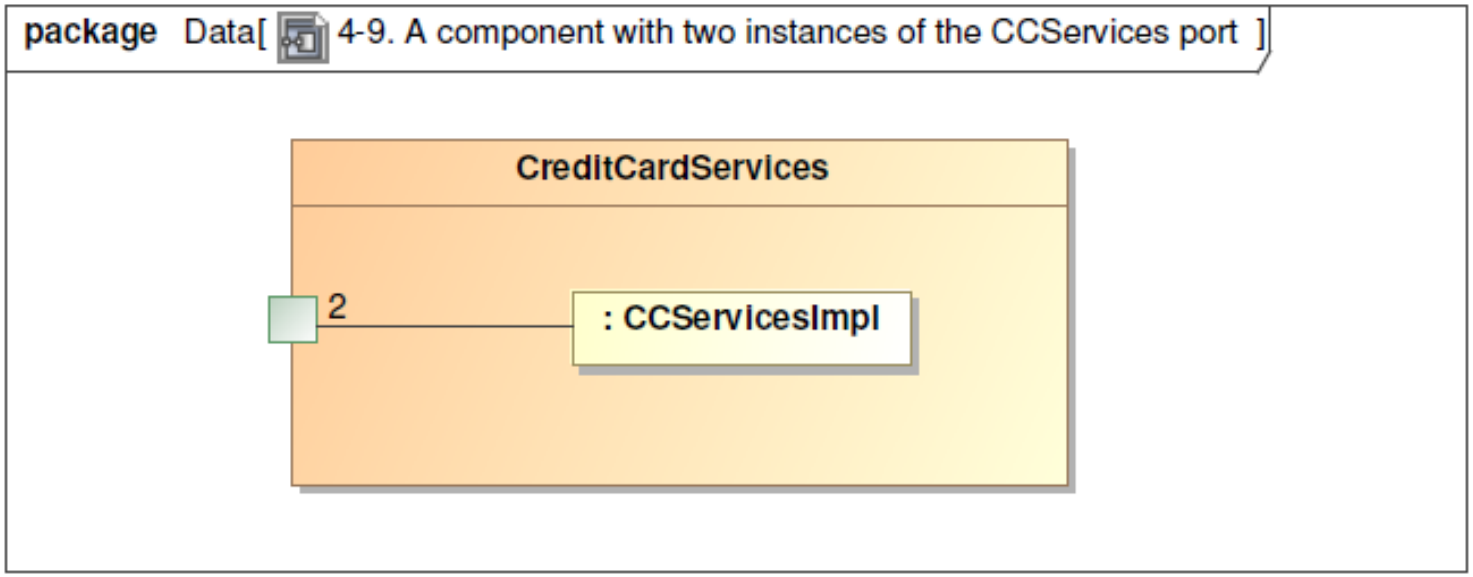
Puertos: comunicación con el exterior (II)

- Los puertos se relacionan con las partes internas que realizan/usan su funcionalidad
  - La realizan en el caso de las interfaces proporcionadas
  - La usan en el caso de las interfaces requeridas

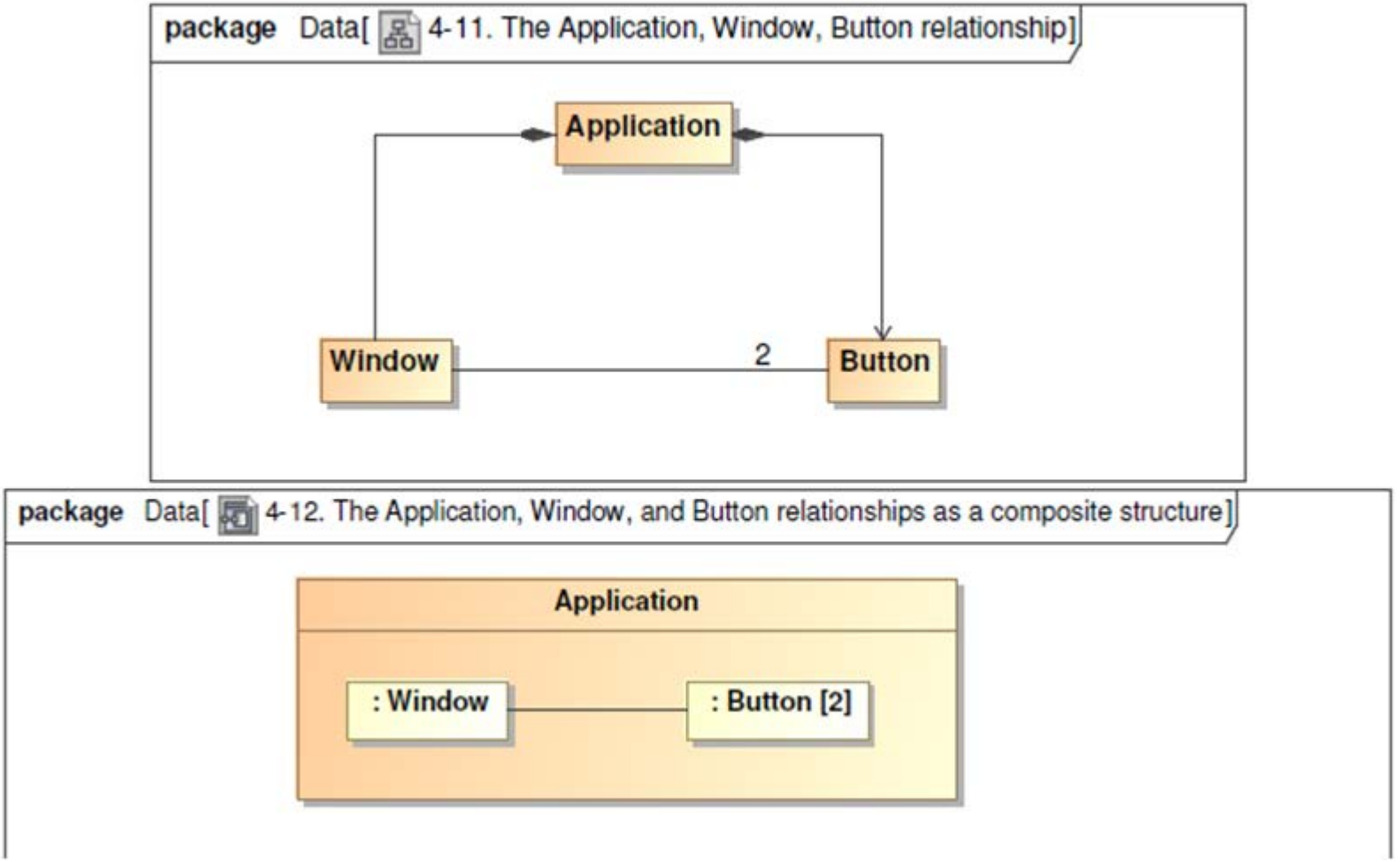


Puertos: comunicación con el exterior (III)

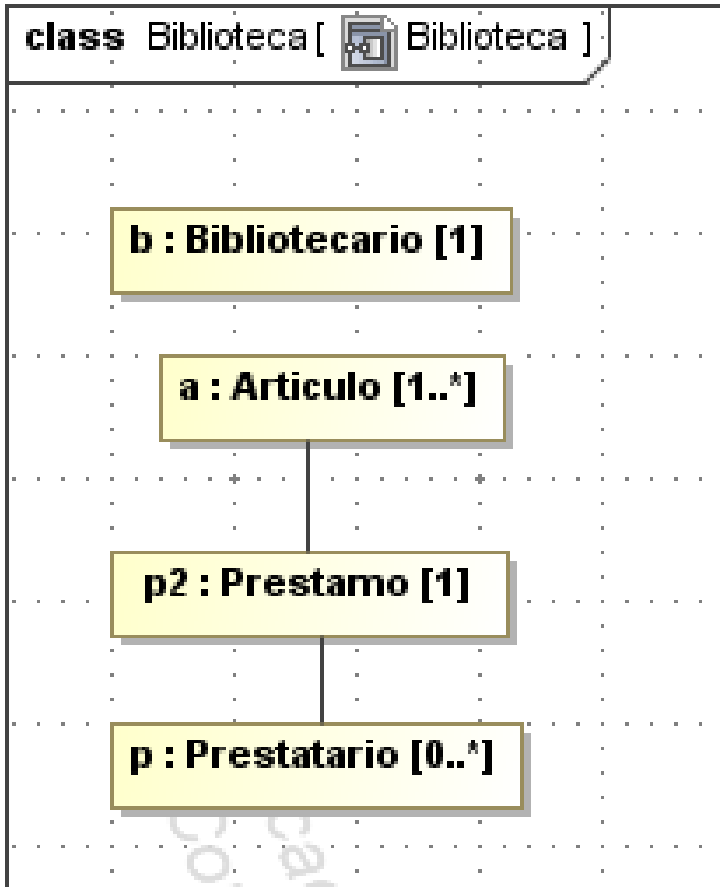
- Los puertos pueden tener multiplicidad
- La estructura compuesta es capaz de distinguir entre los diferentes puertos



Ejemplo


package Data[4-11. The Application, Window, Button relationship]
Application
Window
2
Button
package Data[4-12. The Application, Window, and Button relationships as a composite structure]
Application
: Window
: Button [2]

Ejemplo: Biblioteca



Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

UML : Diagramas de Secuencia

(Diagrama de Comportamiento)

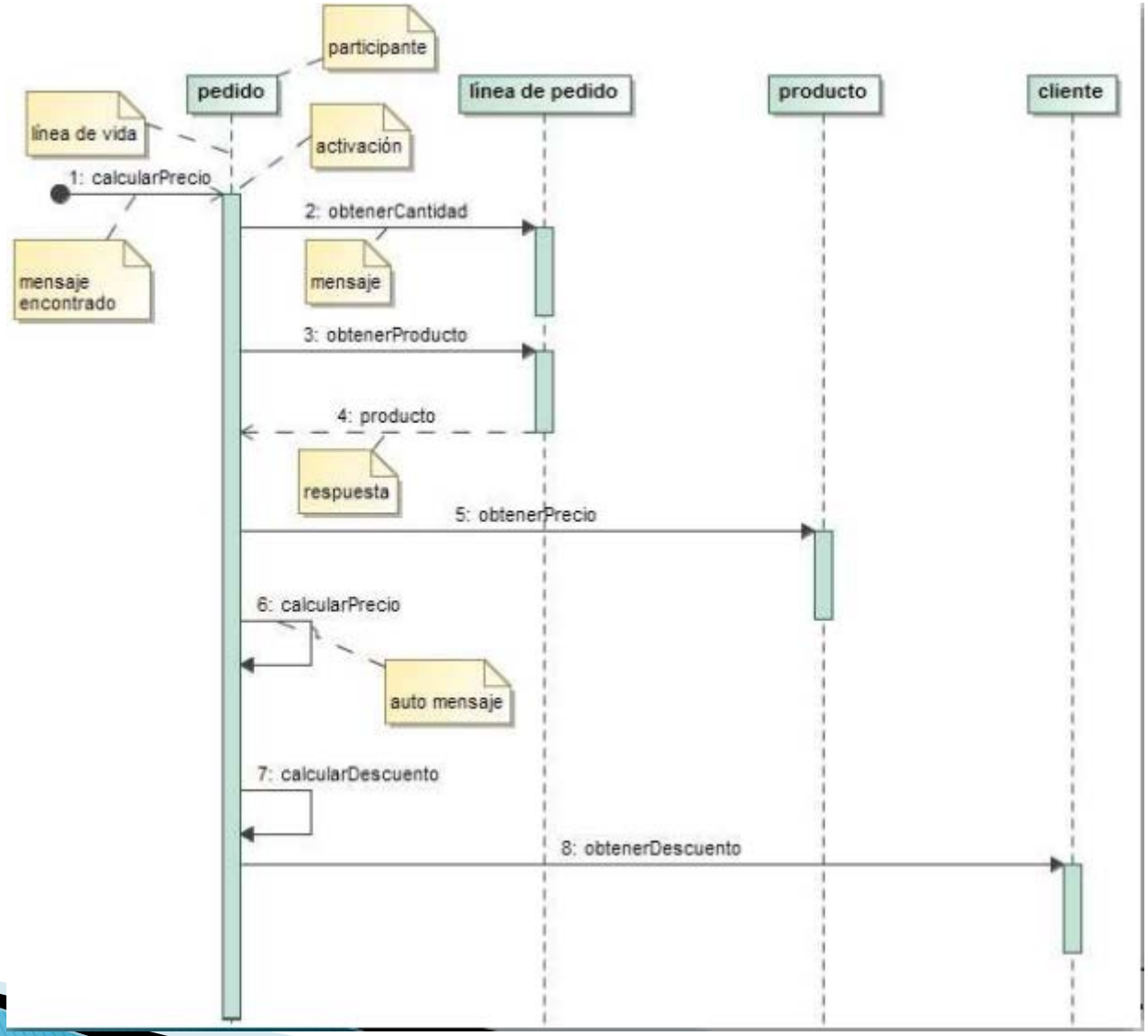
Diagramas de interacción

- Son diagramas de instancias en los que se muestra la interacción entre distintos objetos del sistema
- Diferentes tipos de diagramas de interacción:
  - Diagramas de secuencia
  - Diagramas de colaboración
  - Diagramas temporales
  - Diagramas de vista general de interacción

Diagramas de secuencia

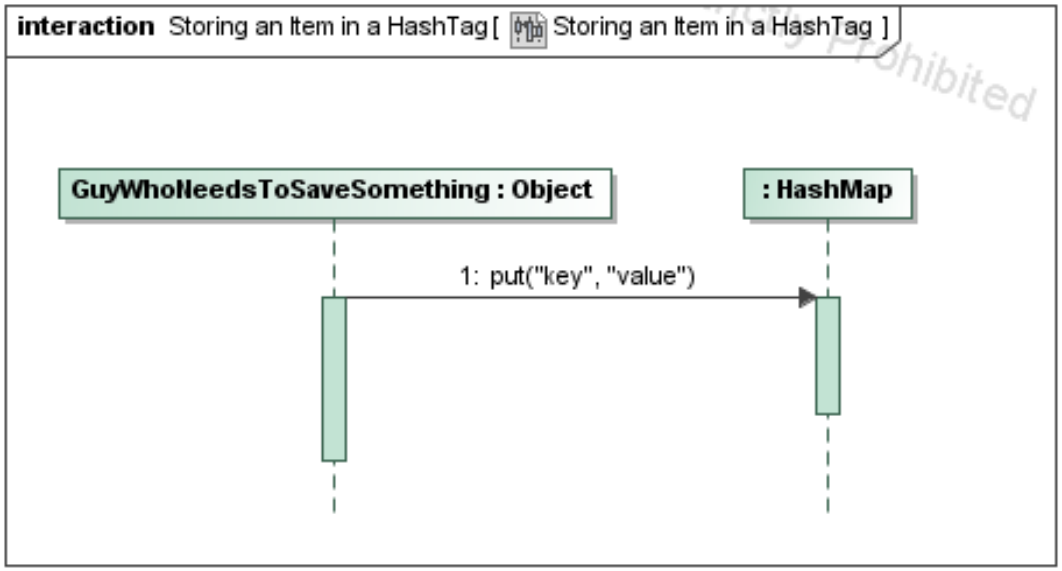
- Muestran la interacción de un conjunto de objetos enfatizando el orden en el tiempo de los mensajes
- En la fase de requisitos, los analistas suelen refinar los casos de usos en uno o mas diagramas de secuencia
  - Contiene detalles de implementación del escenario
    - Objetos y clases usados para la implementación
    - Mensajes intercambiados entre los objetos
- No están pensados para mostrar lógicas de procedimientos complejos

Ejemplo (I)


participante
pedido
línea de pedido
producto
cliente
línea de vida
activación
1: calcularPrecio
2: obtenerCantidad
mensaje encontrado
mensaje
3: obtenerProducto
4: producto
respuesta
5: obtenerPrecio
6: calcularPrecio
auto mensaje
7: calcularDescuento
8: obtenerDescuento

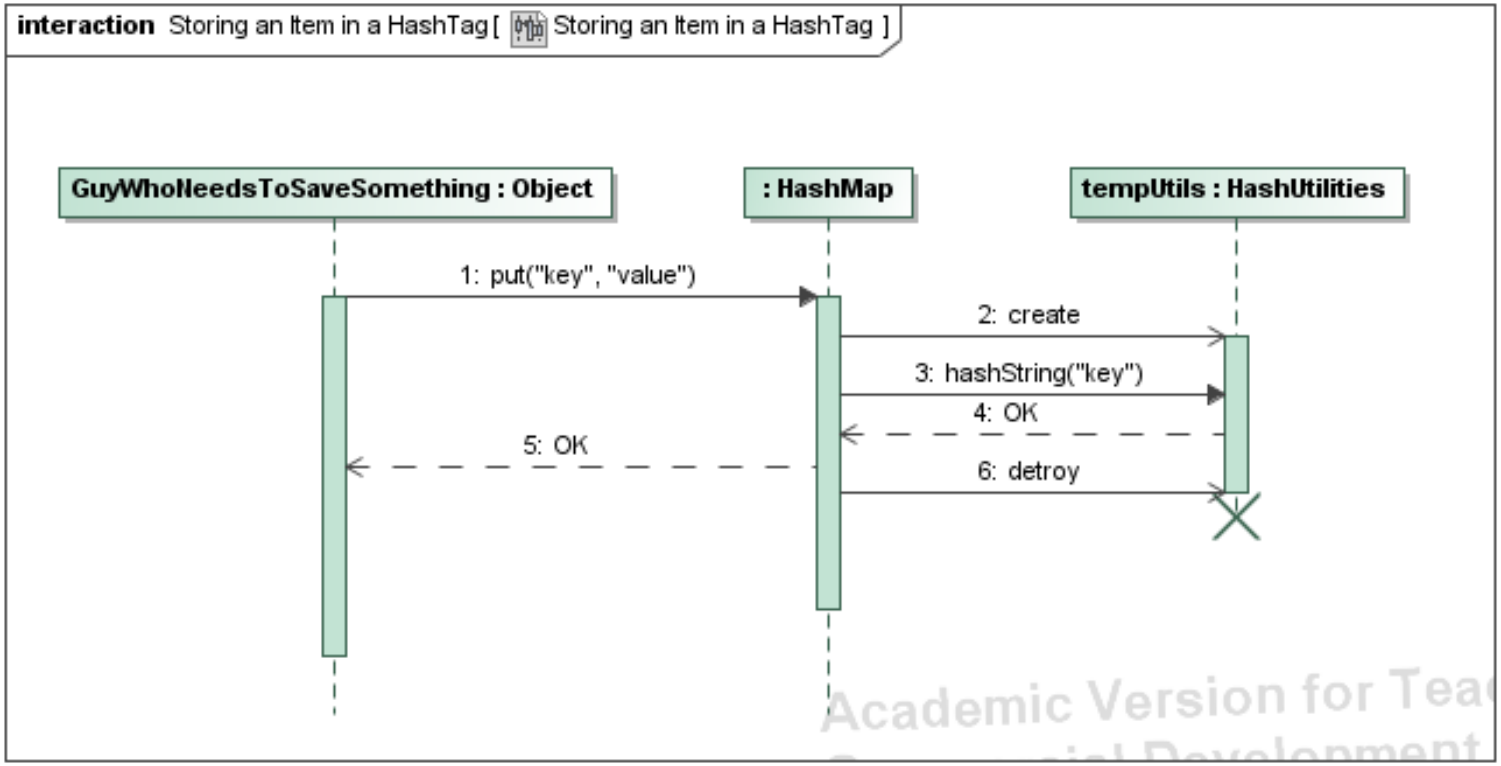
Participantes

- Los participantes son instancias de clases, es decir, objetos
  - Pueden tener nombre o ser anónimos
  - El siguiente diagrama tiene dos participantes, uno llamado *GuyWhoNeedsToSaveSomething* y el otro sin nombre

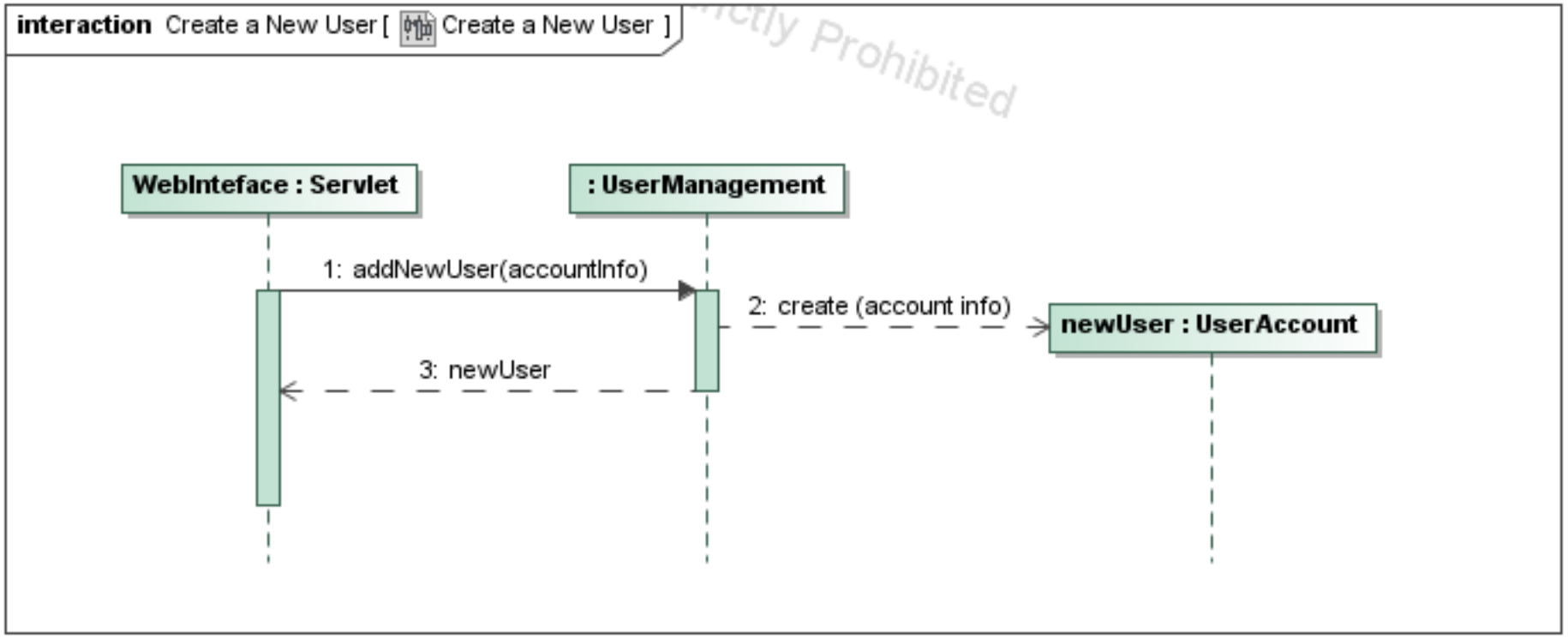


Creación y destrucción (I)

- Los participantes (instancias) se pueden crear y destruir de forma dinámica durante la ejecución

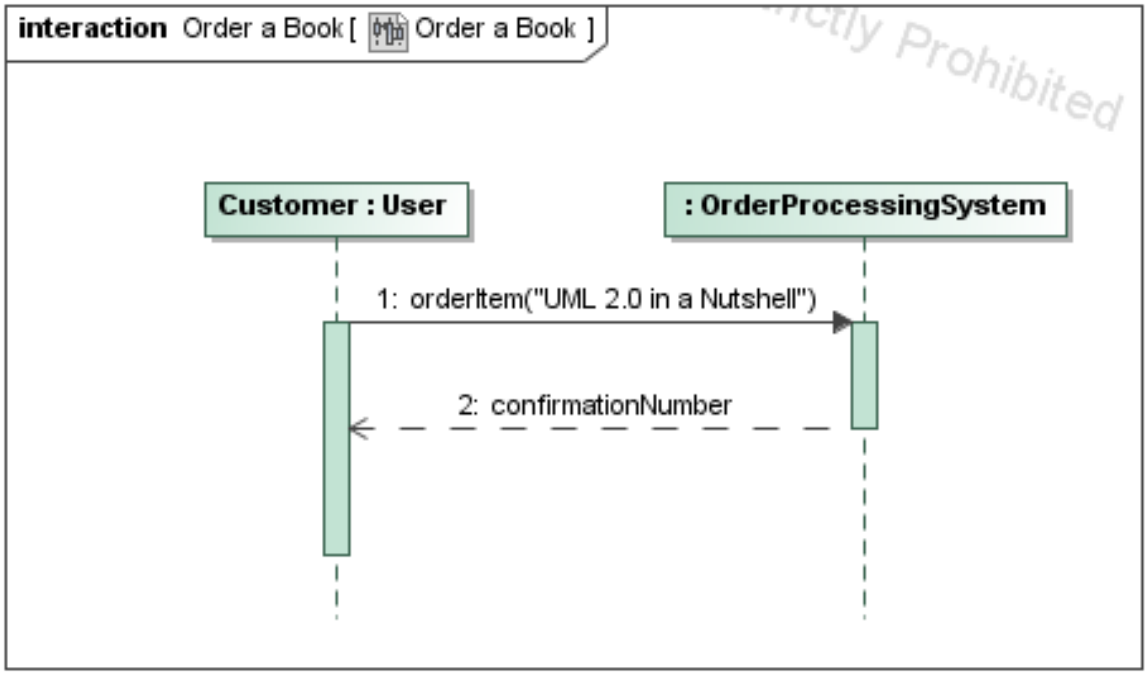


Creación y destrucción (II)



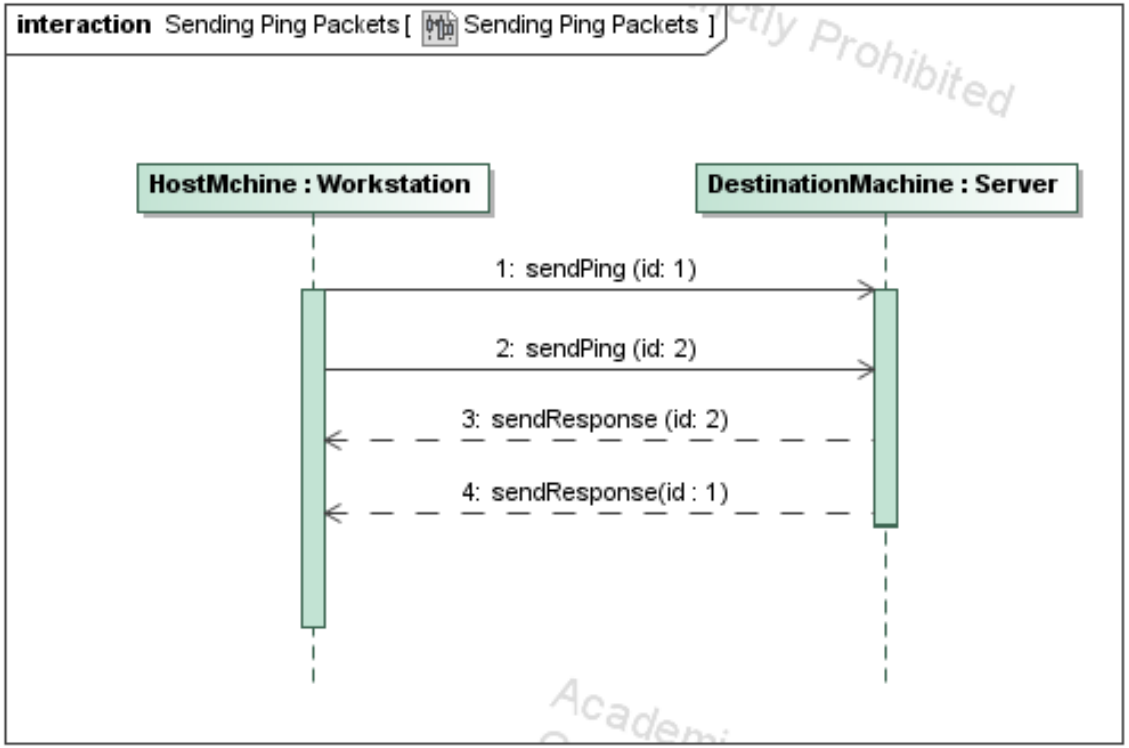
Comunicación síncrona

- El objeto que envía el mensaje queda bloqueado hasta que recibe la respuesta
  - Los mensajes se representan con flechas con la cabeza llena
  - Los mensajes de respuesta se dibujan con línea discontinua



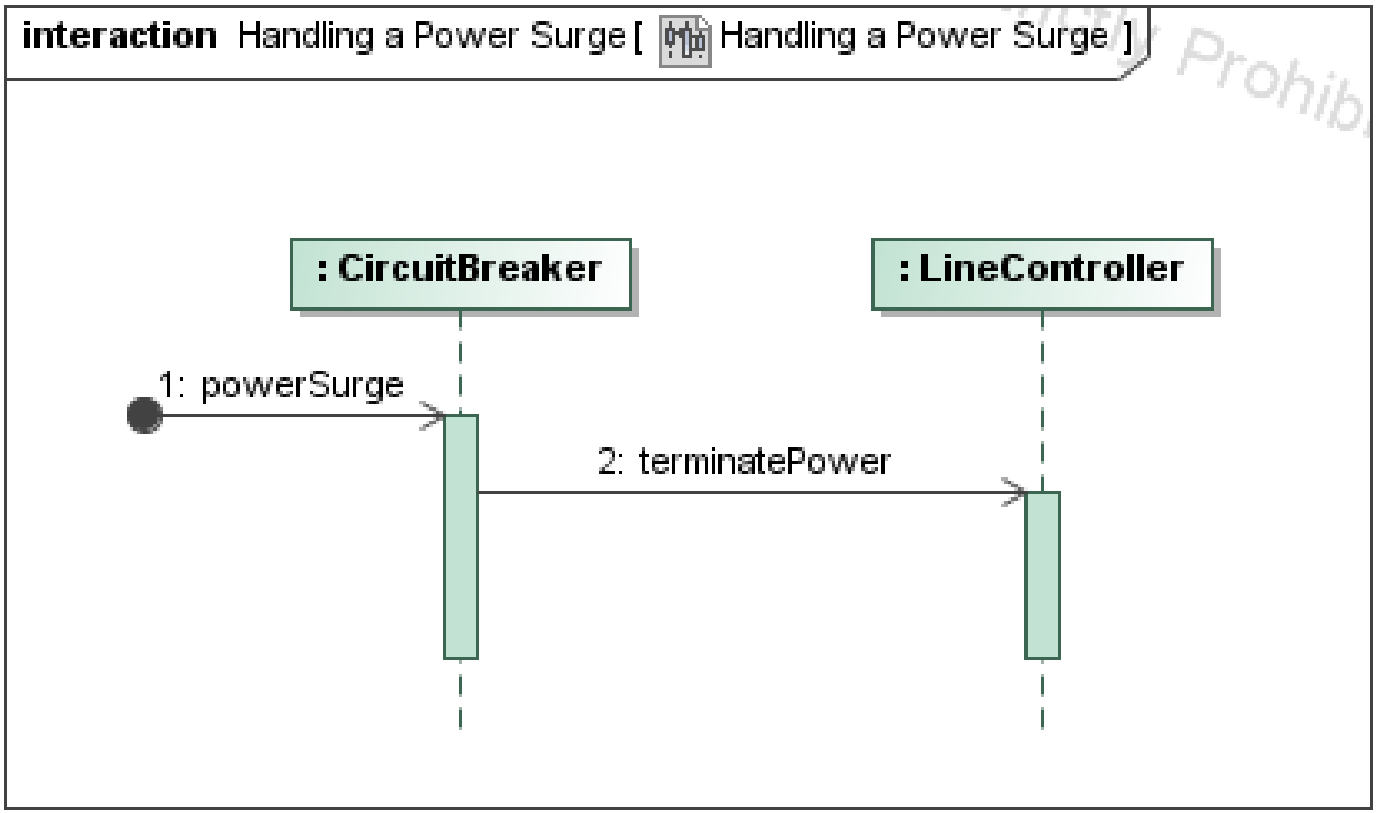
Comunicación asíncrona

- Los mensajes asíncronos terminan inmediatamente
  - Crean un nuevo hilo de ejecución dentro de la secuencia
  - Se representan con flechas con la cabeza abierta



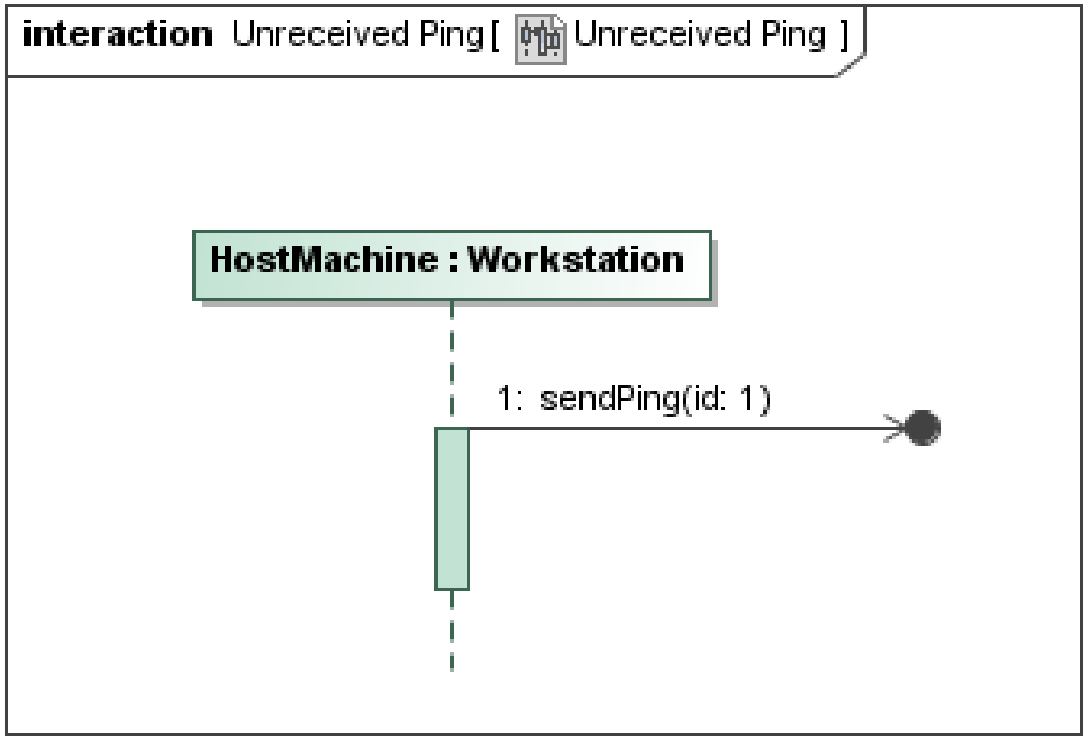
Mensajes encontrados

- Son aquellos cuyo origen no importa



Mensajes perdidos

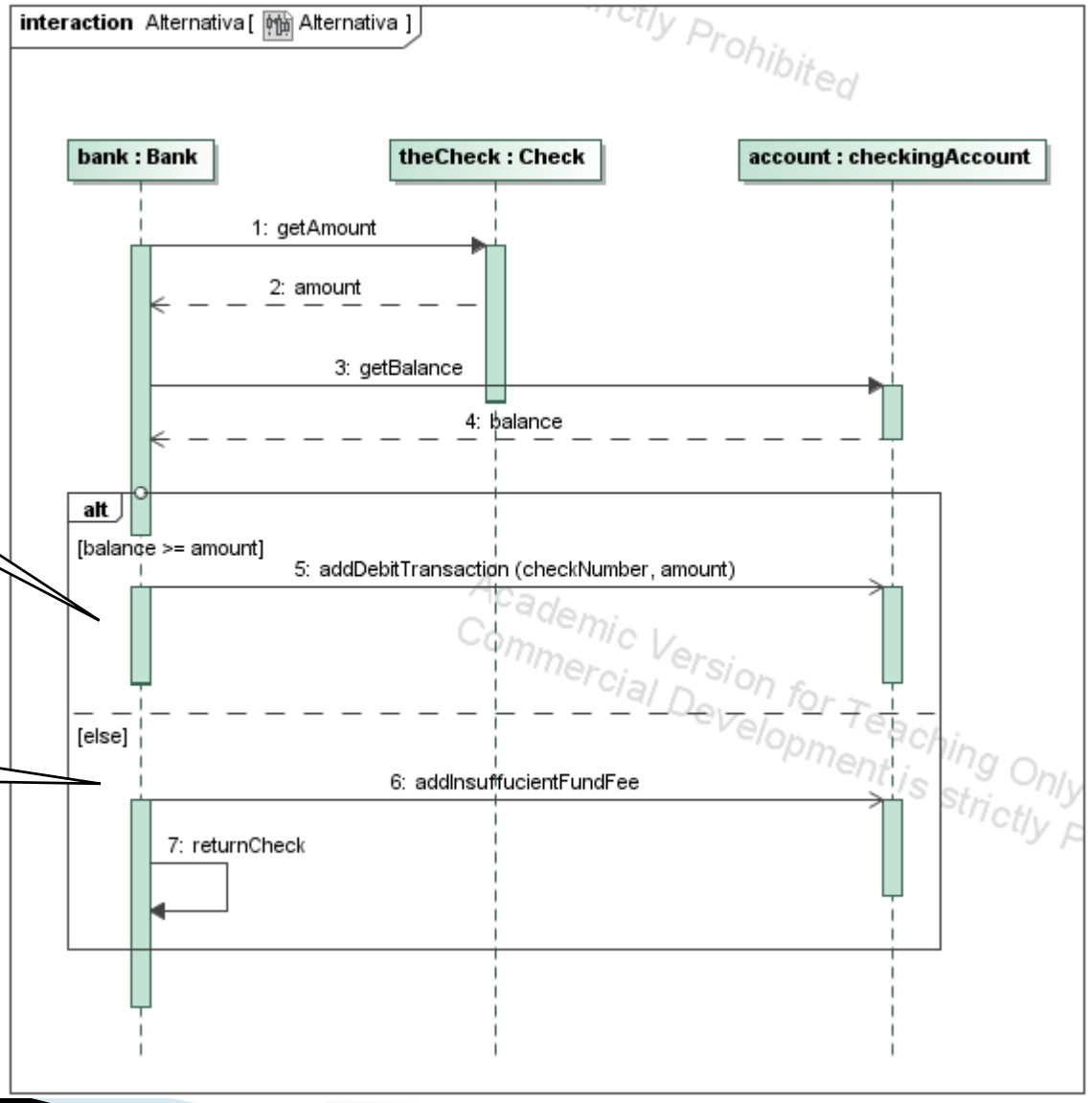
- Son aquellos que no llegan a ningún participante



Fragmentos combinados

- Permiten definir una interacción compleja mediante componentes más simples
  - Alternativa
  - Bucles
  - Opción
  - Break
  - Paralelo
  - Región crítica
  - …

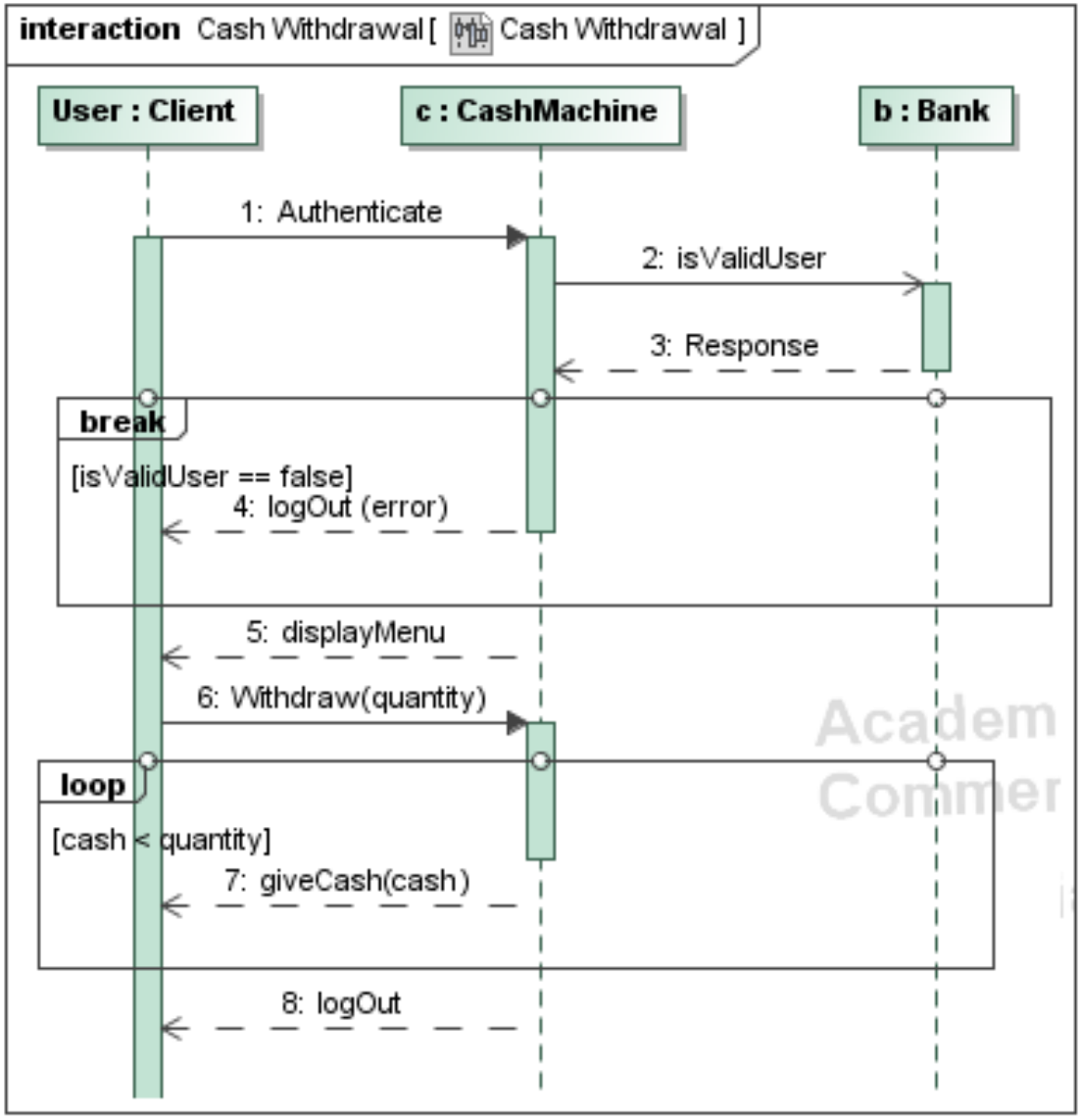
Alternativas



Realiza esto …

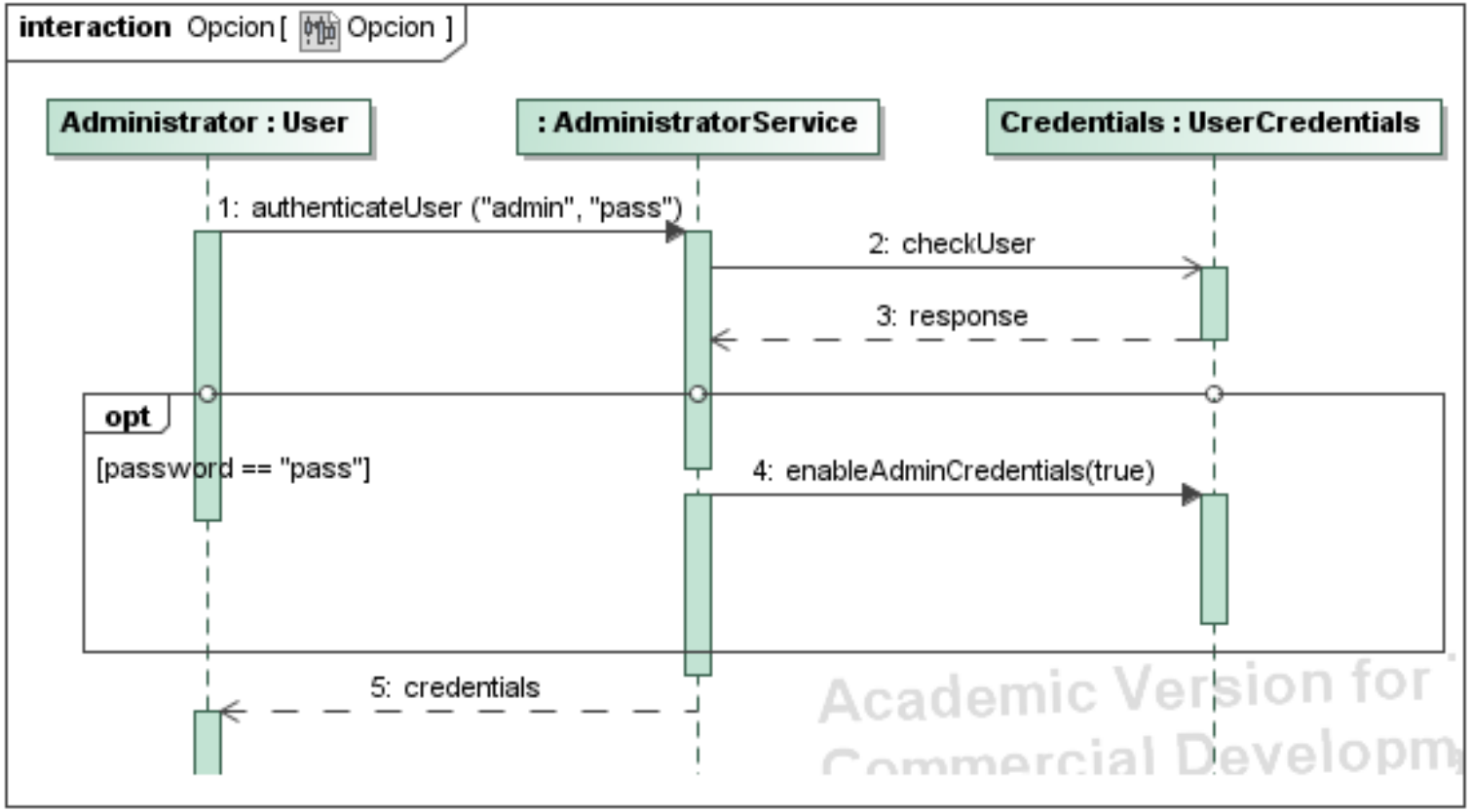
… o realiza esto

Bucles


interaction Cash Withdrawal [Cash Withdrawal]
User : Client
c : CashMachine
b : Bank
1: Authenticate
2: isValidUser
3: Response
break
[isValidUser == false]
4: logOut (error)
5: displayMenu
6: Withdraw(quantity)
loop
[cash < quantity]
7: giveCash(cash)
8: logOut

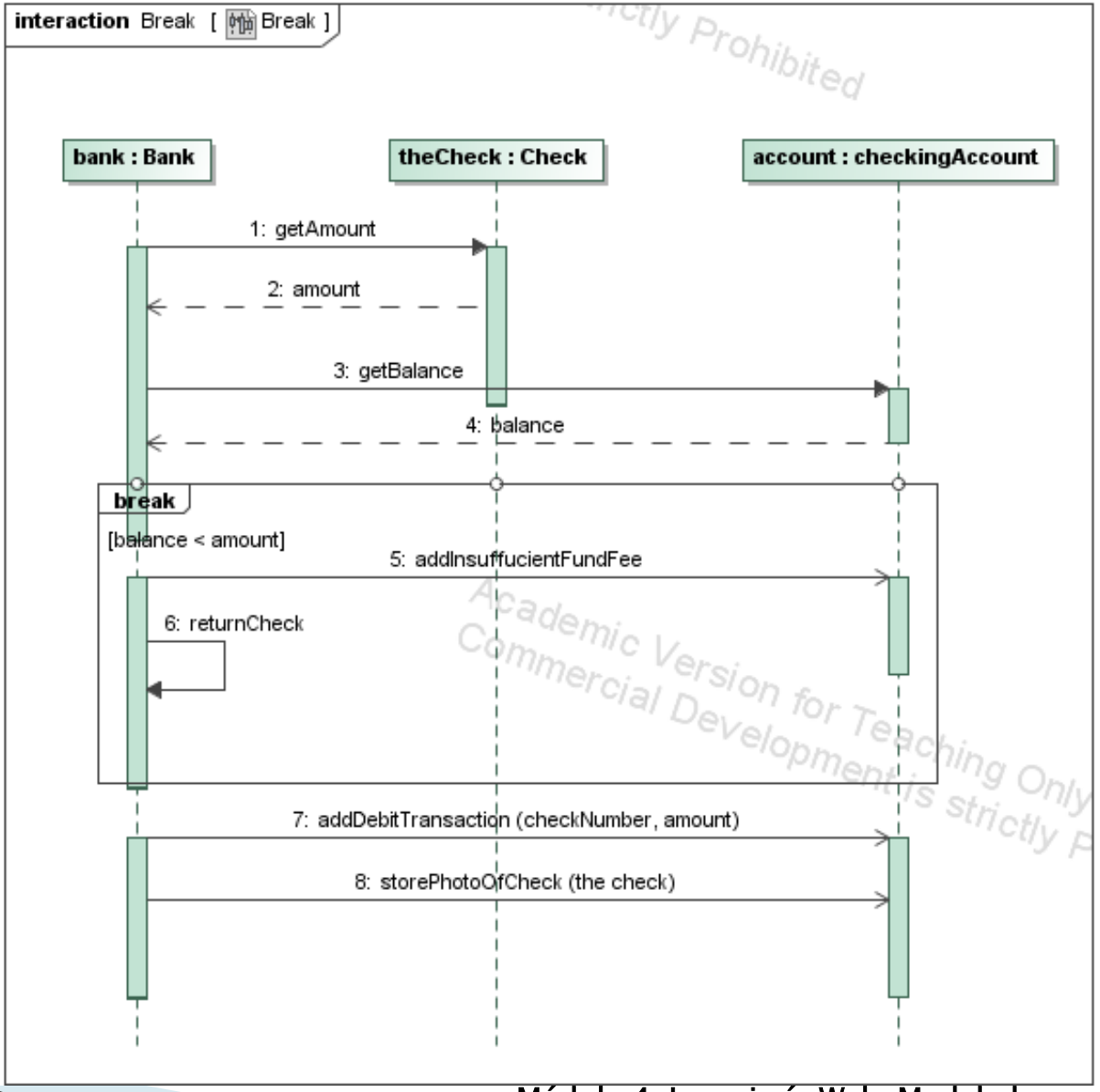
Opción

- Ciertas acciones se ejecutan sólo si se cumple la condición



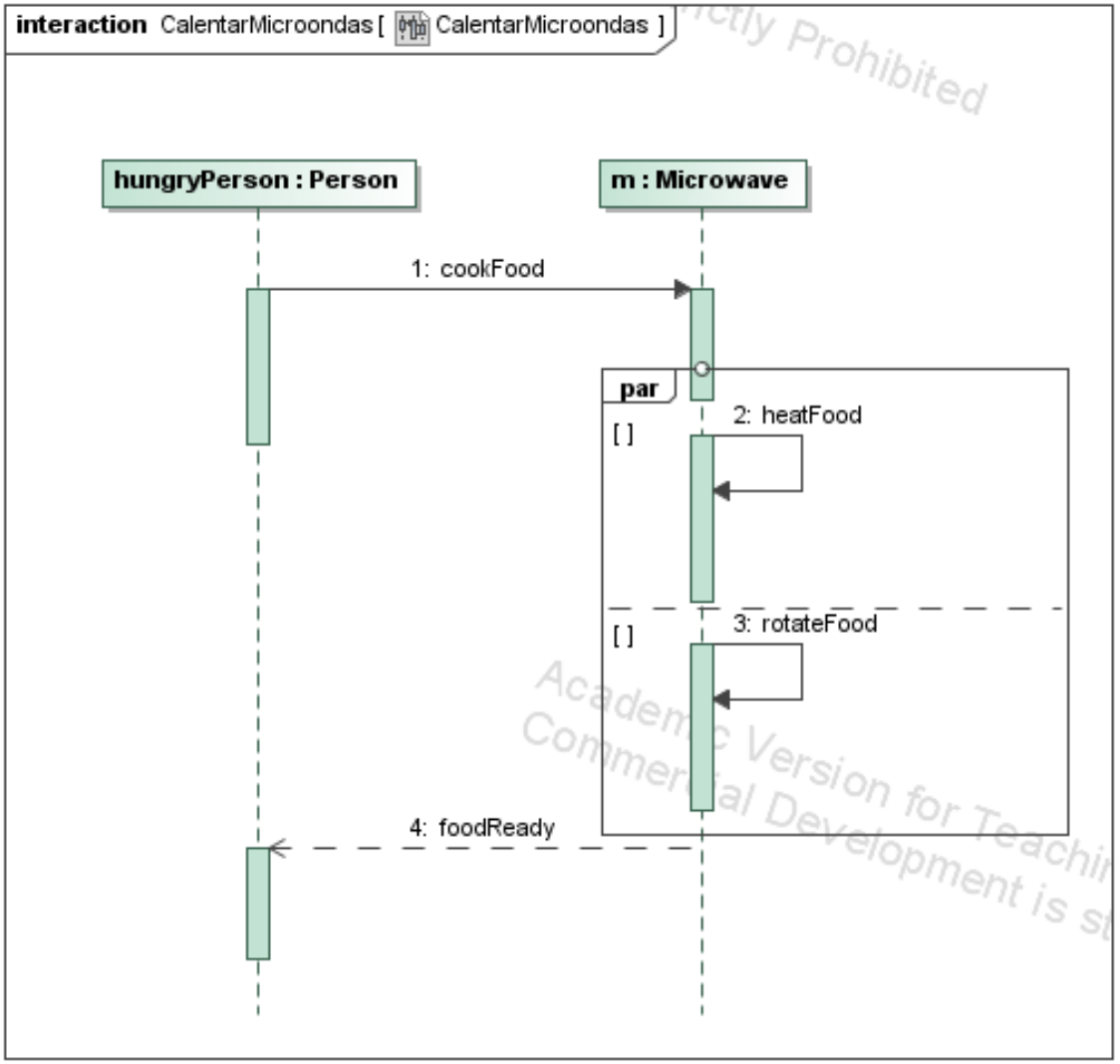
Break

- Como la opción, pero lo que sigue no se ejecuta si entra en el *break*



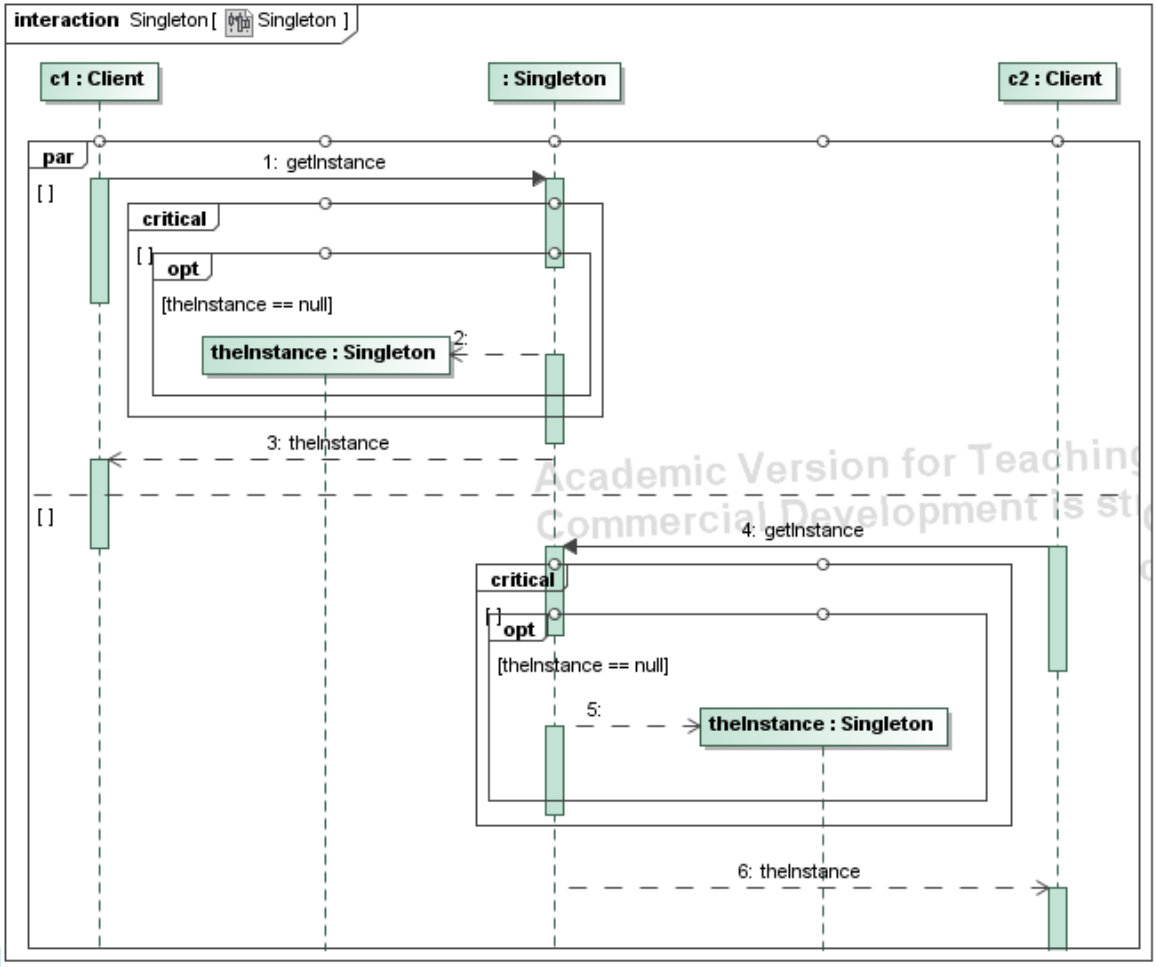
Paralelo

- Especifica que varias acciones se realizan a la vez, cada una en un hilo de ejecución



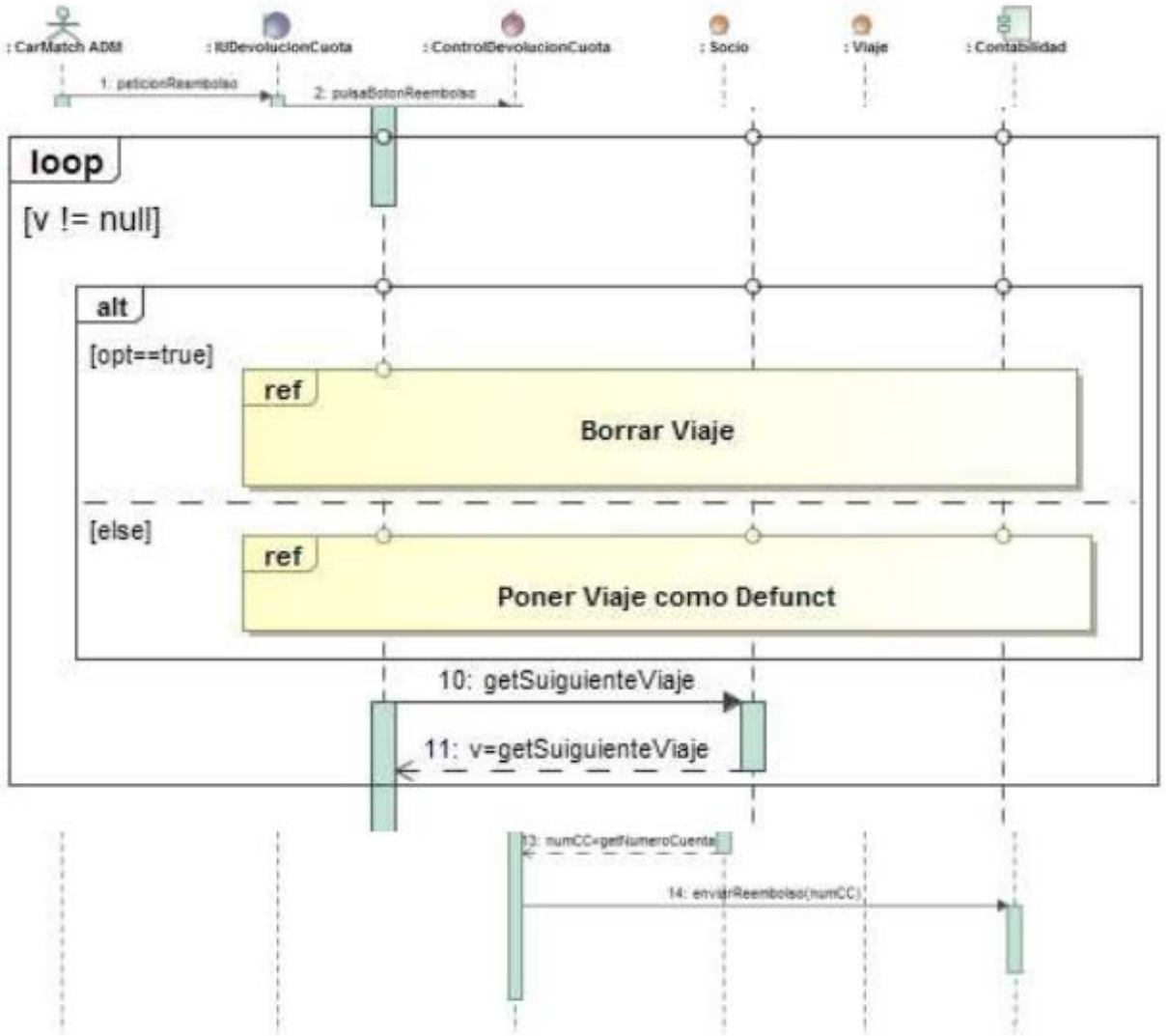
Región crítica

- El fragmento crítico sólo puede tener un hilo de ejecución y debe ejecutarse de una vez

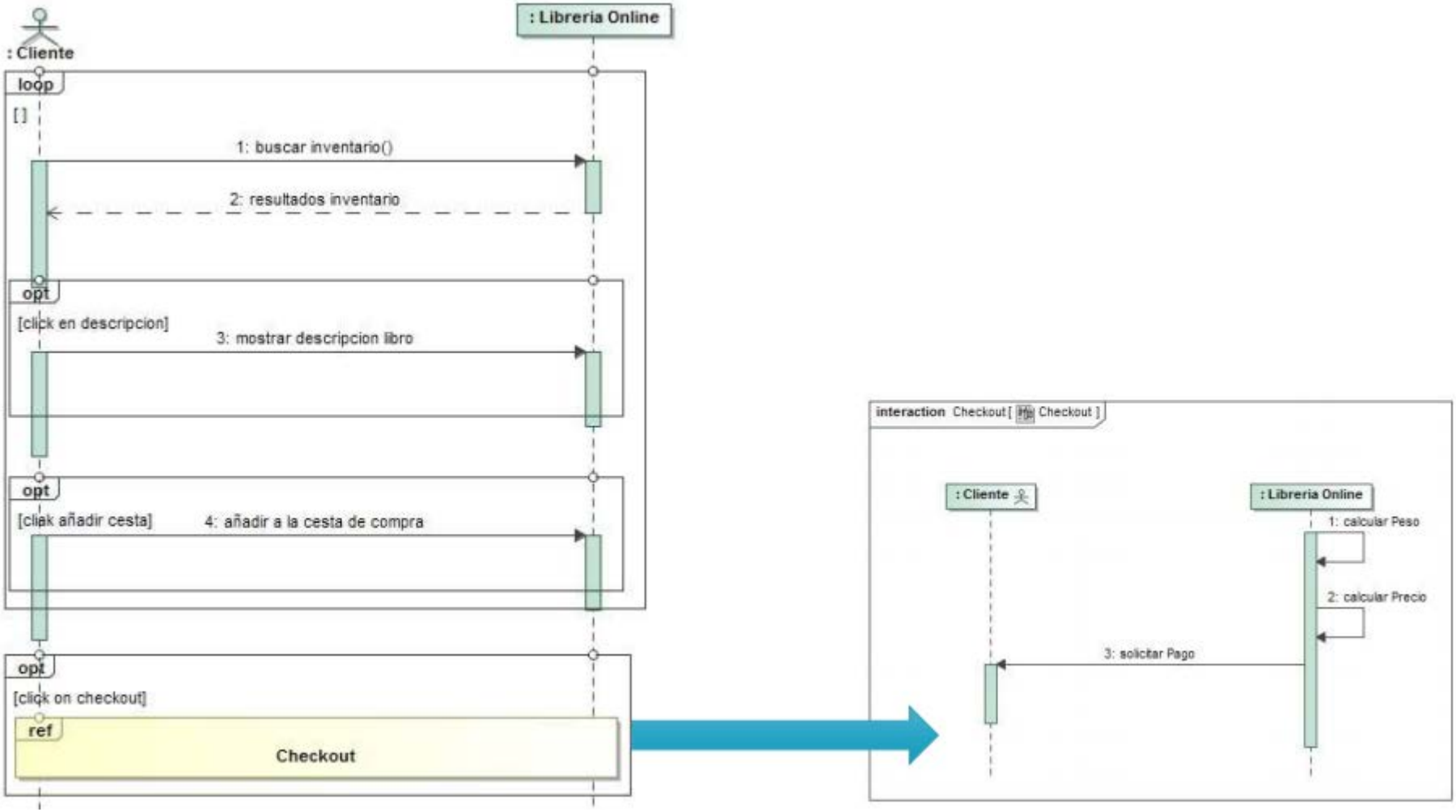


Referencias (I)

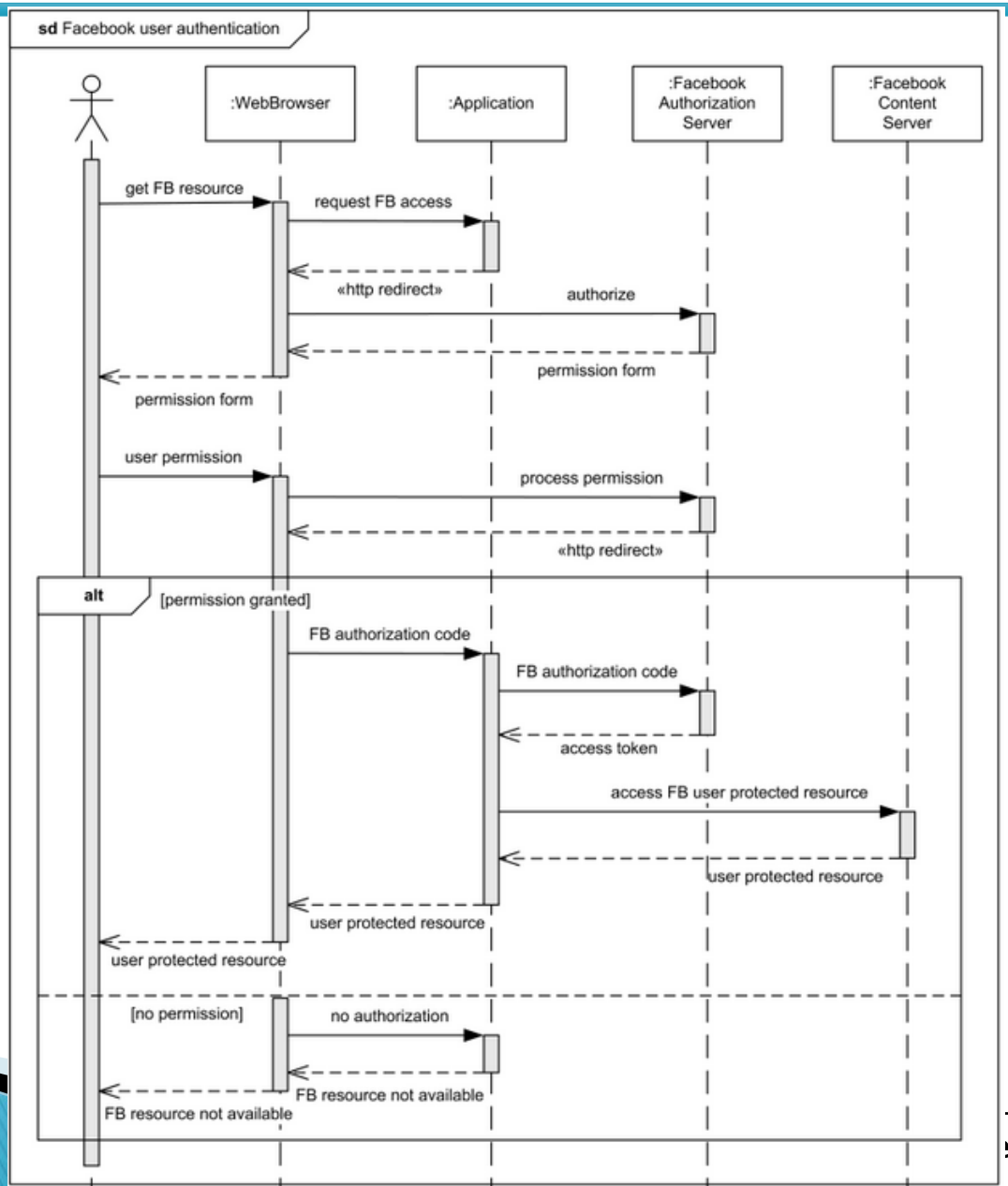
- Una referencia a otro diagrama de secuencia
- Puede tener parámetros



Referencias - Diseño de alto nivel



Autenticación en Facebook



Ejemplo: Factoría de martillos

- El siguiente proceso se repite mientras que el número de martillos recolectados por el Usuario sea menor que un *máximo*:
  - El Generador de mangos y el Generador de cabezas depositan un mango y una cabeza, respectivamente, en el Ensamblador. Ambas máquinas trabajan en paralelo.
  - A continuación, el Ensamblador deposita un martillo en la Bandeja, y hace saber a ambos Generadores que tiene espacio disponible
  - La Bandeja le hace saber al Usuario que hay un martillo disponible, y el Usuario lo recoge. La Bandeja le hace saber al Ensamblador que tiene espacio disponible

Ejercicio: Préstamo de artículo

- Préstamo de artículo
  - El Prestatario solicita un préstamo al Bibliotecario
    - El Bibliotecario consulta con la Biblioteca si el usuario está identificado
      - Si no lo está, el Bibliotecario le dice al Prestatario que no está identificado y se acaba el proceso
      - Si está identificado, el Bibliotecario consulta a la Biblioteca si se está dentro de plazo para préstamos
        - Si se está dentro de plazo, el Bibliotecario presta el libro al Prestatario
        - Si se está fuera de plazo, no le presta el libro

Ejercicio: Devolución de artículo

- Devolución de artículo
  - El Prestatario solicita al Bibliotecario la devolución de un artículo
    - El Bibliotecario comprueba la fecha límite de devolución del artículo con la Biblioteca
      - Si la fecha es posterior al día de hoy, el Bibliotecario acepta la devolución del artículo
      - Si no, le Bibliotecario le pregunta a la Biblioteca la cuantía de la sanción al Prestatario
        - Si la sanción es mayor que 500, el Bibliotecario suspende y sanciona al Prestatario
        - Si es menor, sólo le sanciona

Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

Perfiles UML

Índice

- Introducción
- Mecanismos de extensión de UML
- Perfiles UML

Un ejemplo

- Queremos realizar un modelado de datos **con UML**
- Modelado de datos: es el proceso de crear un modelo de datos para un sistema de información aplicando técnicas formales de modelado de datos
  - Desarrollo de software para gestión de un gimnasio
  - Nos centramos en el modelado de las estructuras de datos involucradas en la aplicación
  - Posibles modelos:
    - Modelo conceptual,
    - Modelo lógico,
    - Modelo físico

Modelado de datos

- En nuestro ejemplo nos vamos a fijar en dos modelos
  - Modelo del dominio (esquema conceptual)
    - Describe la semántica del dominio, es decir, el ámbito del modelo
    - Define lo que se puede hacer en el dominio en un "lenguaje artificial"
    - Es el primer paso al organizar los requisitos de datos
    - Similar a un diagrama de clases
  - Modelo físico
    - Describe los medios físicos usados para almacenar los datos

Modelado de datos: modelo del dominio

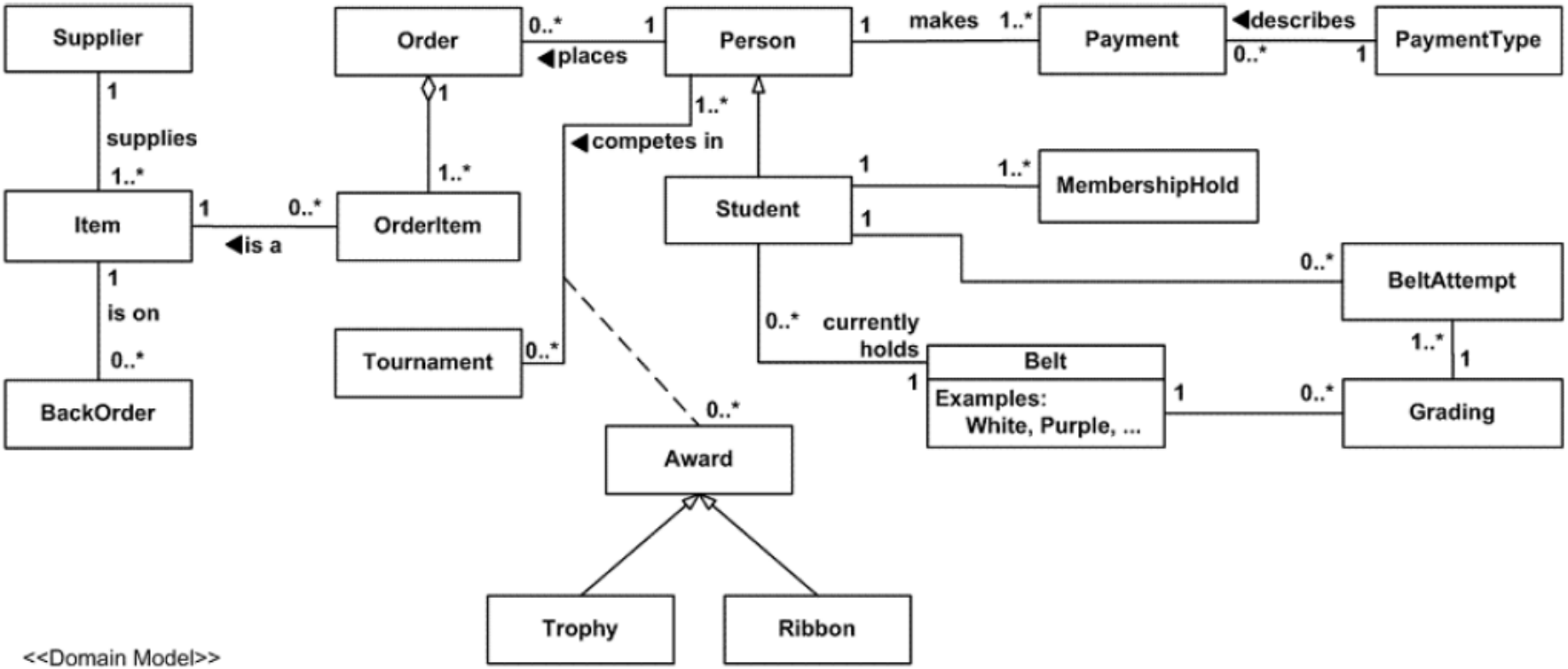


Imagen tomada de *http://www.agiledata.org/essays/agileDataModeling.html*

Modelado de datos: modelo físico

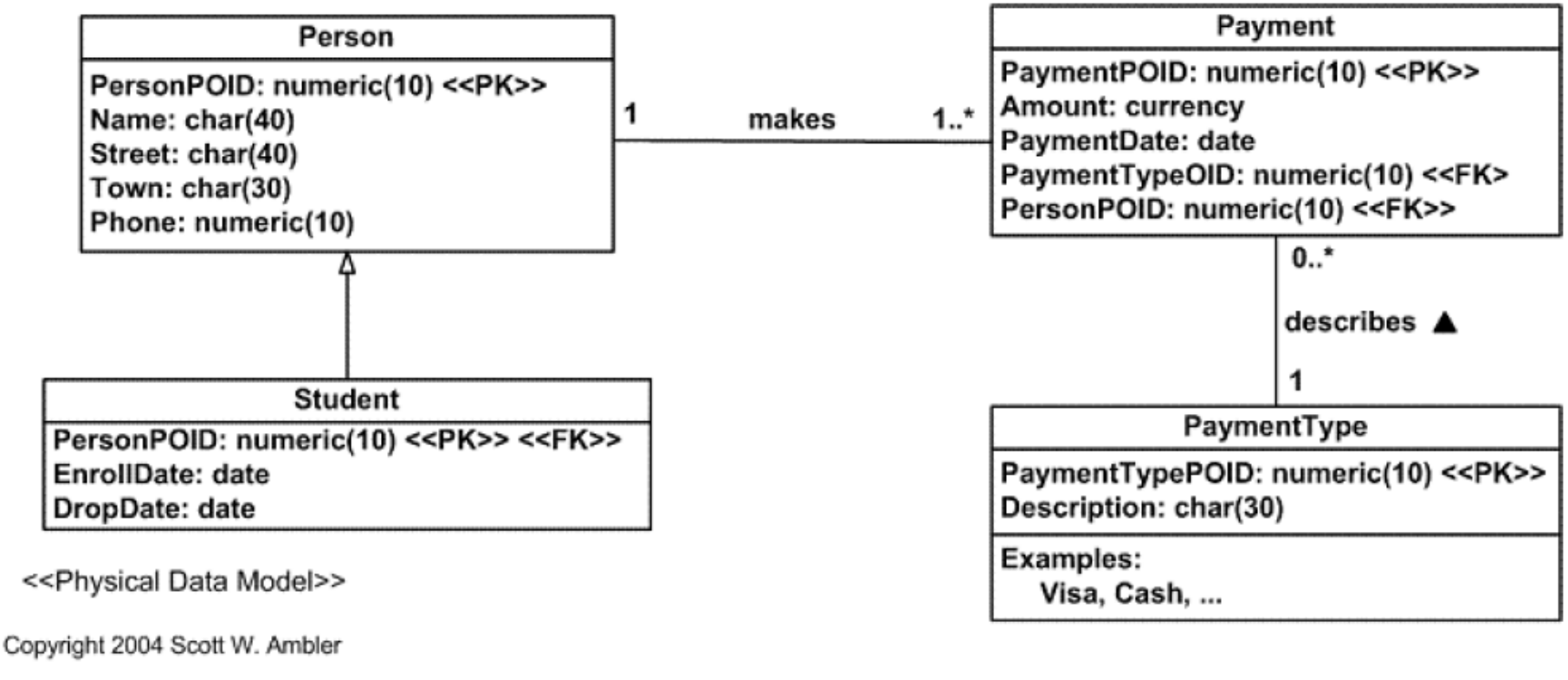


Imagen tomada de *http://www.agiledata.org/essays/agileDataModeling.html*

Modelado de datos

- Necesitamos distinguir varios aspectos:
  - Un diagrama representa el **modelo del dominio** y otro el **modelo físico.**
    - El modelo del dominio se puede especificar prácticamente con un diagrama de clases
    - En el modelo físico, algunos atributos representan **claves primarias** y otros representan **claves externas**
      - UML estándar no dispone de elementos para distinguir dichos aspectos
      - Necesitamos elementos extras en UML

Índice

- Introducción
- Mecanismos de extensión de UML
- Perfiles UML

Mecanismos de extensión de UML

"El hecho de que UML sea un lenguaje de propósito general proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas.

Sin embargo, hay numerosas ocasiones en las que es mejor contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares.

Esto sucede, por ejemplo, cuando la sintaxis o la semántica de UML no permiten expresar los conceptos específicos del dominio,

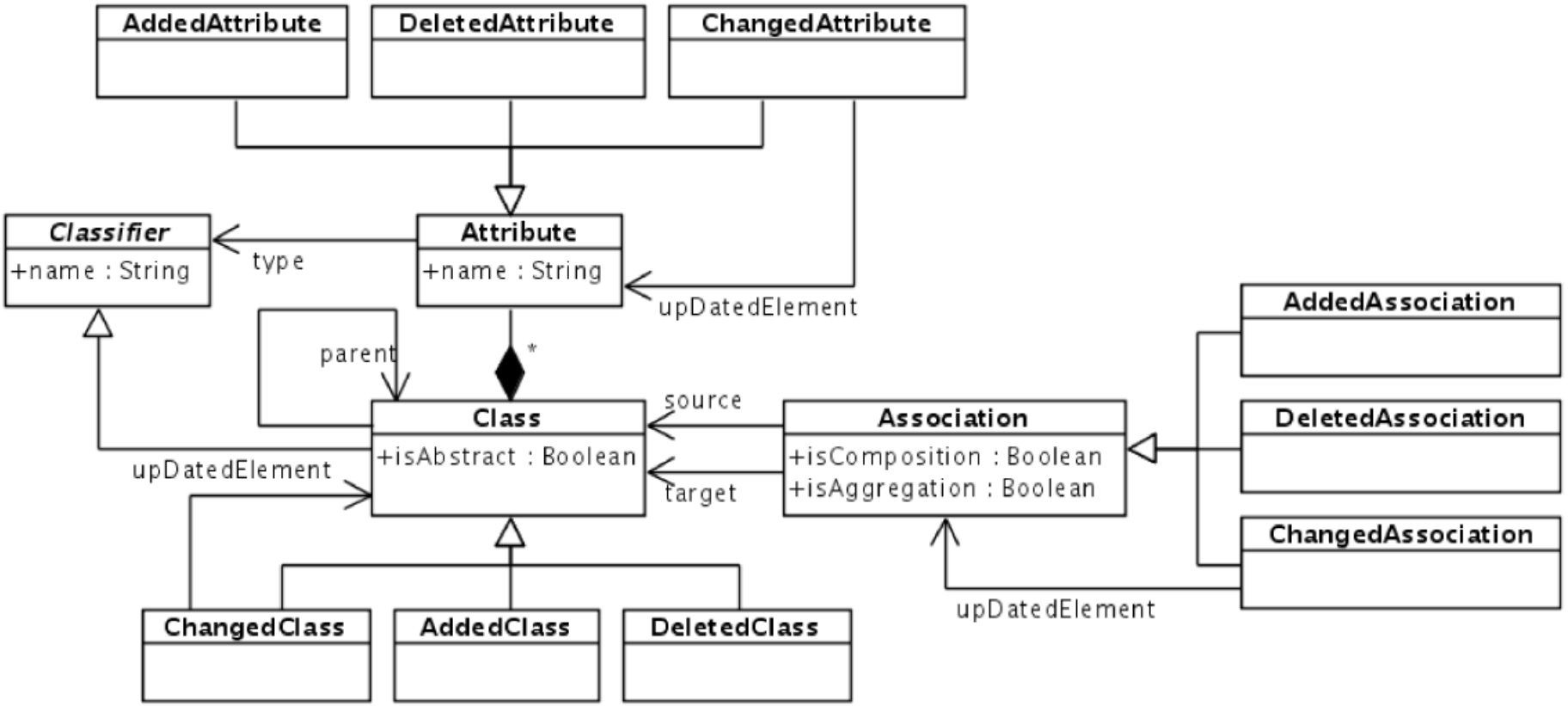
o cuando se desea restringir y especializar los constructores propios de UML, que suelen ser demasiado genéricos y numerosos."

Una Introducción a los Perfiles UML. Lidia Fuentes y Antonio Vallecillo

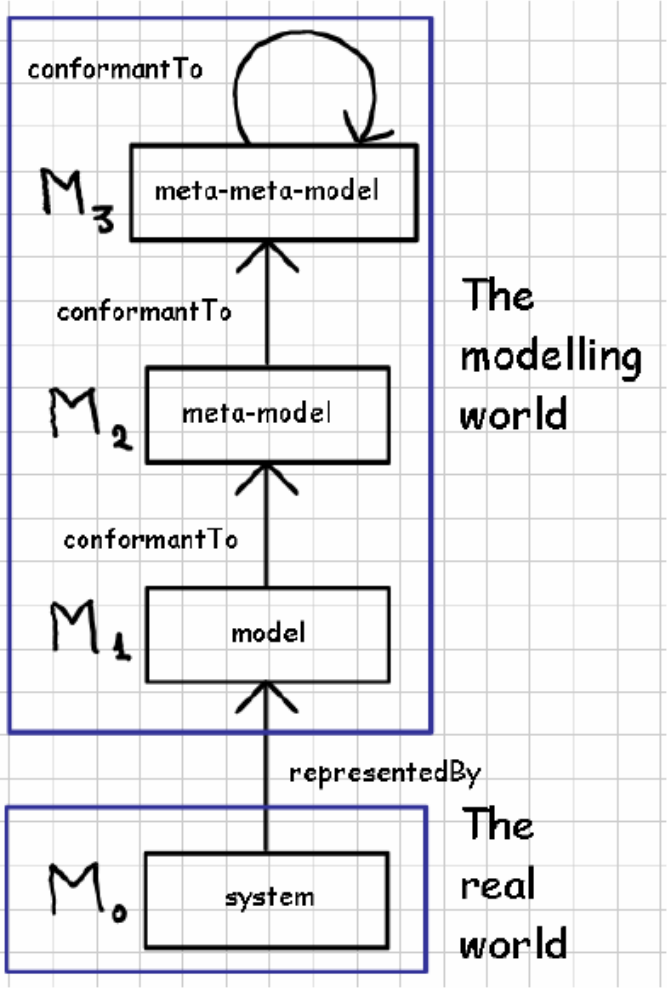
Mecanismos de extensión de UML

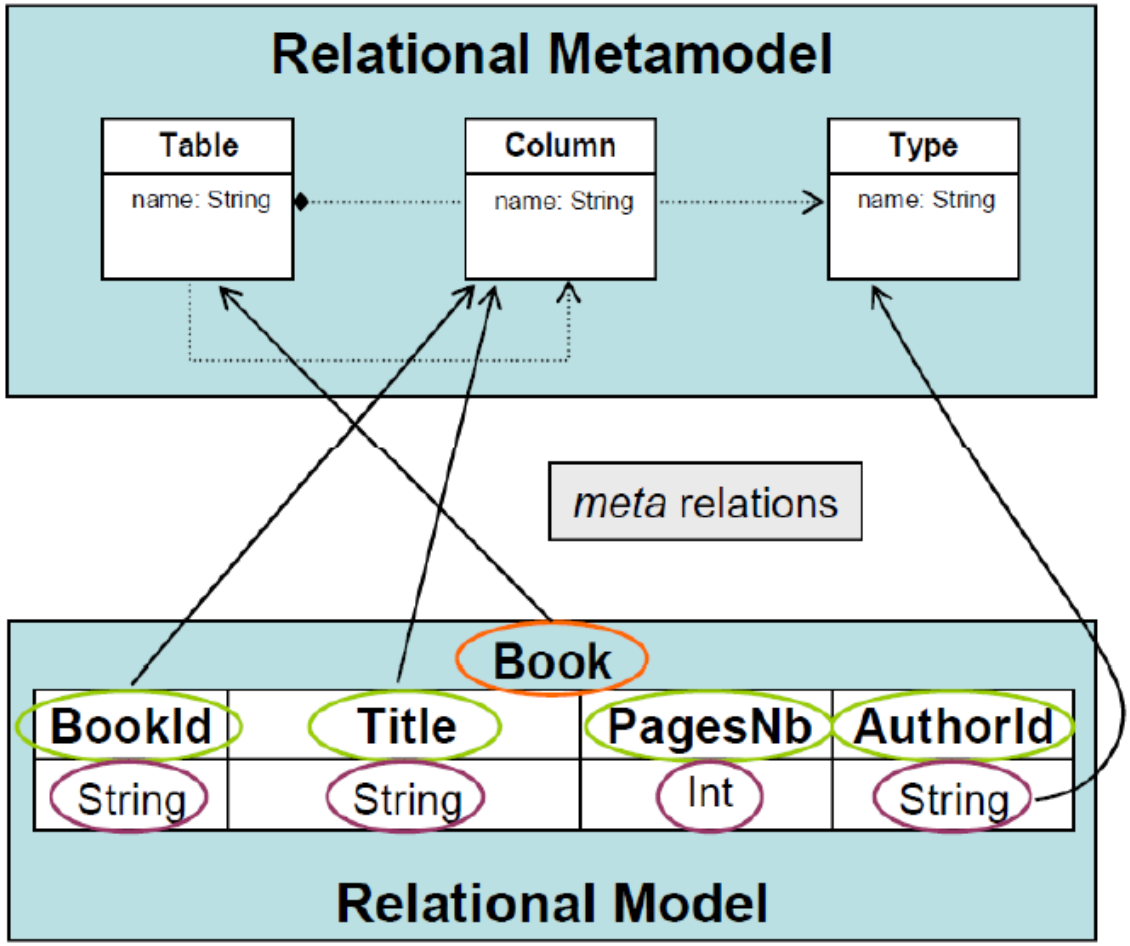
- Creación de un nuevo lenguaje a partir de MOF
  - MOF es un lenguaje simple para definir lenguajes como UML
  - Ejemplo: CWM
    - CWM es una especificación para describir objetos y relaciones en el contexto de los almacenes de datos
    - Incoveniente: no es compatible con UML
- Ampliación de UML creando nuevas clases en el metamodelo usando MOF
  - Inconveniente: UML cada vez es más grande con elementos pocos usados
- Creación de perfiles: adaptación de elementos de UML con nuevas propiedades

Parte del metamodelo de UML

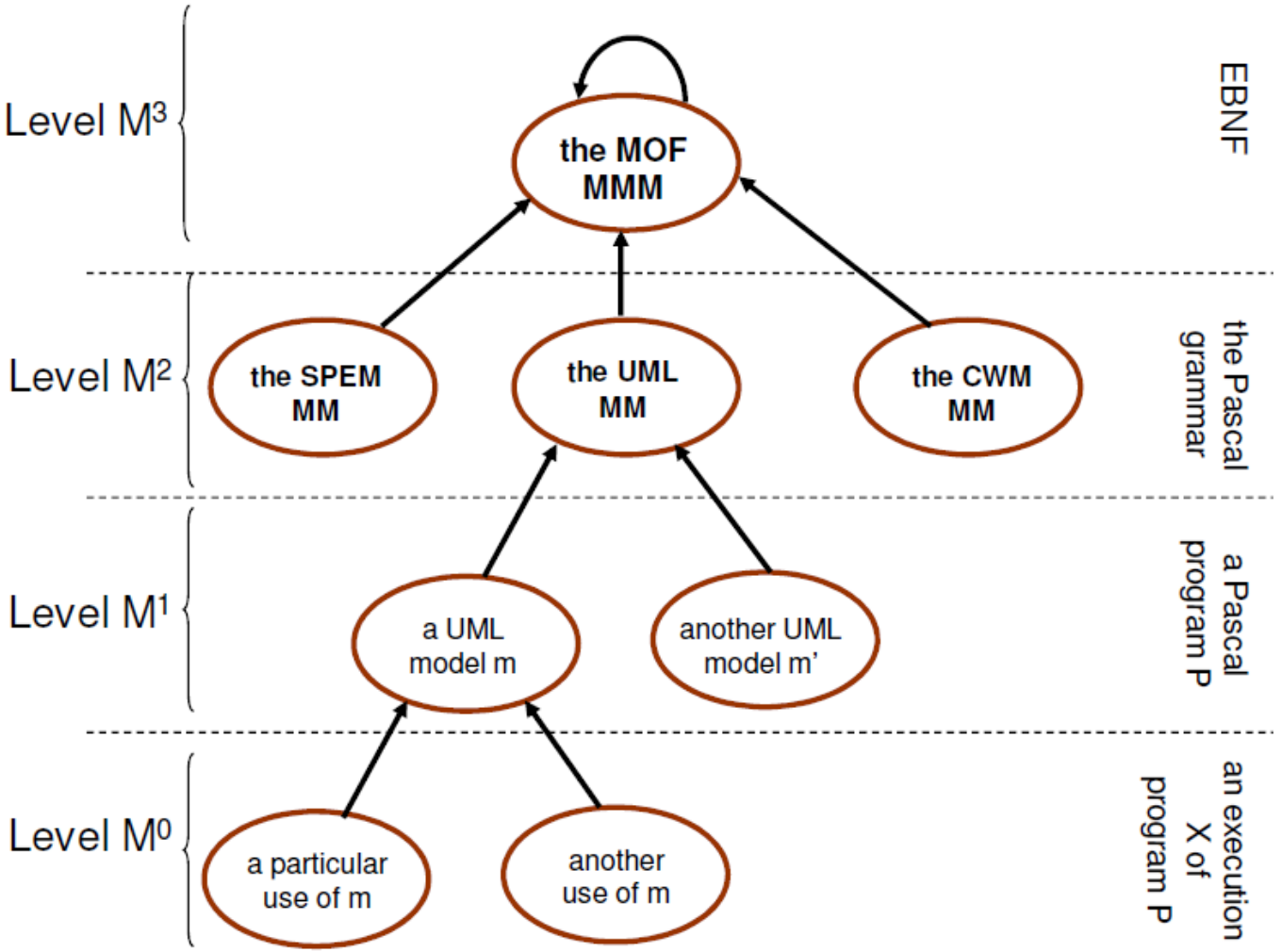


Niveles de metamodelado

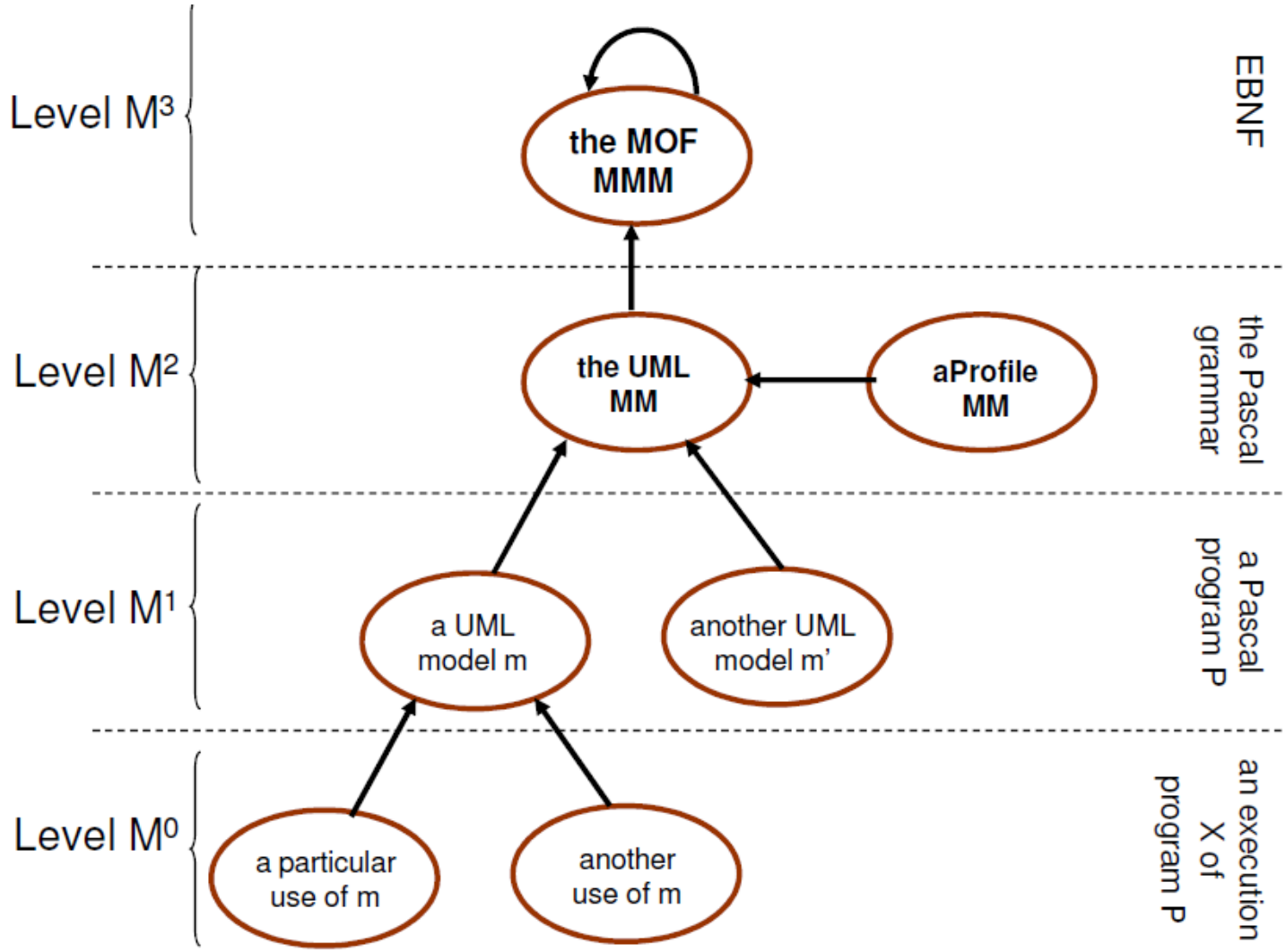




Creación de nuevos lenguajes a partir de MOF



Creación de perfiles de UML



Índice

- Introducción
- Mecanismos de extensión de UML
- Perfiles UML

¿Cuándo crear un nuevo perfil?

Razones por las que un diseñador puede querer extender y adaptar un metamodelo existente:

- Disponer de una **terminología y vocabulario propio** de un dominio de aplicación o de una plataforma de implementación concreta (por ejemplo, poder manejar dentro del modelo del sistema terminología propia de EJB como “Home Interface”, “archive”, etc)
  - Los EJB, Enterprise JavaBeans, son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Oracle Corporation
- Definir una **sintaxis para construcciones** que no cuentan con una notación propia (como sucede con las acciones)
- Definir una **nueva notación para símbolos ya existentes**, más acorde con el dominio de la aplicación objetivo (poder usar, por ejemplo, una figura con un ordenador en lugar del símbolo para representar un nodo que por defecto ofrece UML para representar ordenadores en una red)
- Añadir cierta **semántica que no aparece de forma precisa** en el metamodelo (por ejemplo, la incorporación de prioridades en la recepción de señales en una máquina de estados de UML)
- Añadir cierta **semántica que no existe en el metamodelo** (por ejemplo, relojes, tiempo continuo, etc)
- Añadir **restricciones** a las existentes en el metamodelo, restringiendo su forma de utilización (por ejemplo, impidiendo que ciertas acciones se ejecuten en paralelo dentro de una transición, o forzando la existencia de ciertas asociaciones entre las clases del modelo

Una Introducción a los Perfiles UML. Lidia Fuentes y Antonio Vallecillo

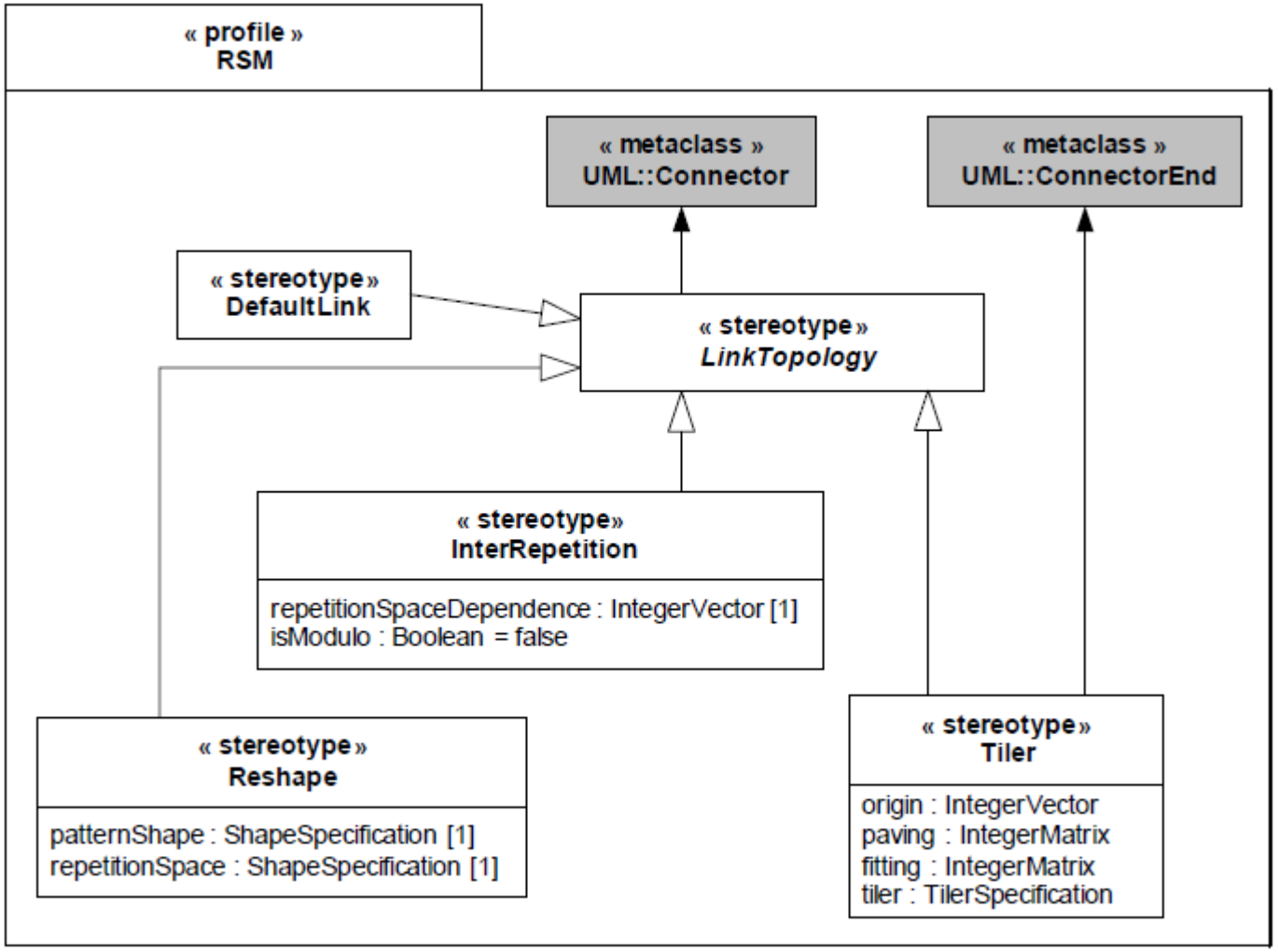
Perfiles estándar de UML

- UML Profile for Enterprise Distributed Object Computing (EDOC)
- UML Profile for Modeling QoS and Fault Tolerance
- Characteristics and Mechanisms, UML Profile for Schedulability, Performance and Time
- Service oriented architecture Modeling Language (SoaML)
- UML Profile for Systems Engineering (SysML)
- UML Testing Profile
- UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE)

http://www.omg.org/technology/documents/profile_catalog.htm

MARTE

- Ejemplo de un extracto de MARTE



Elementos de un perfil

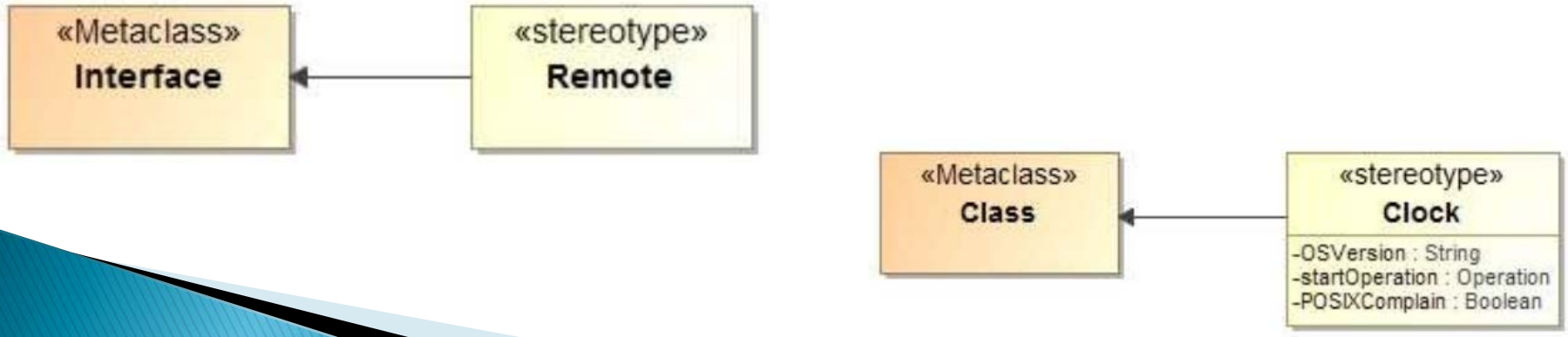
- Un perfil UML se compone de tres mecanismos de extensión
  - Estereotipos
  - Definiciones de etiqueta
  - Restricciones

Estereotipos

- La forma de generar un elemento nuevo con nuevas propiedades es aplicar un **estereotipo** a un elemento de UML ya existente
  - Un estereotipo tiene nombre
  - Sólo se puede aplicar a un conjunto definido de elementos de UML, que se indican en la definición del estereotipo
  - La forma de representarlo es marcar la nueva clase con el símbolo <<stereotype>>

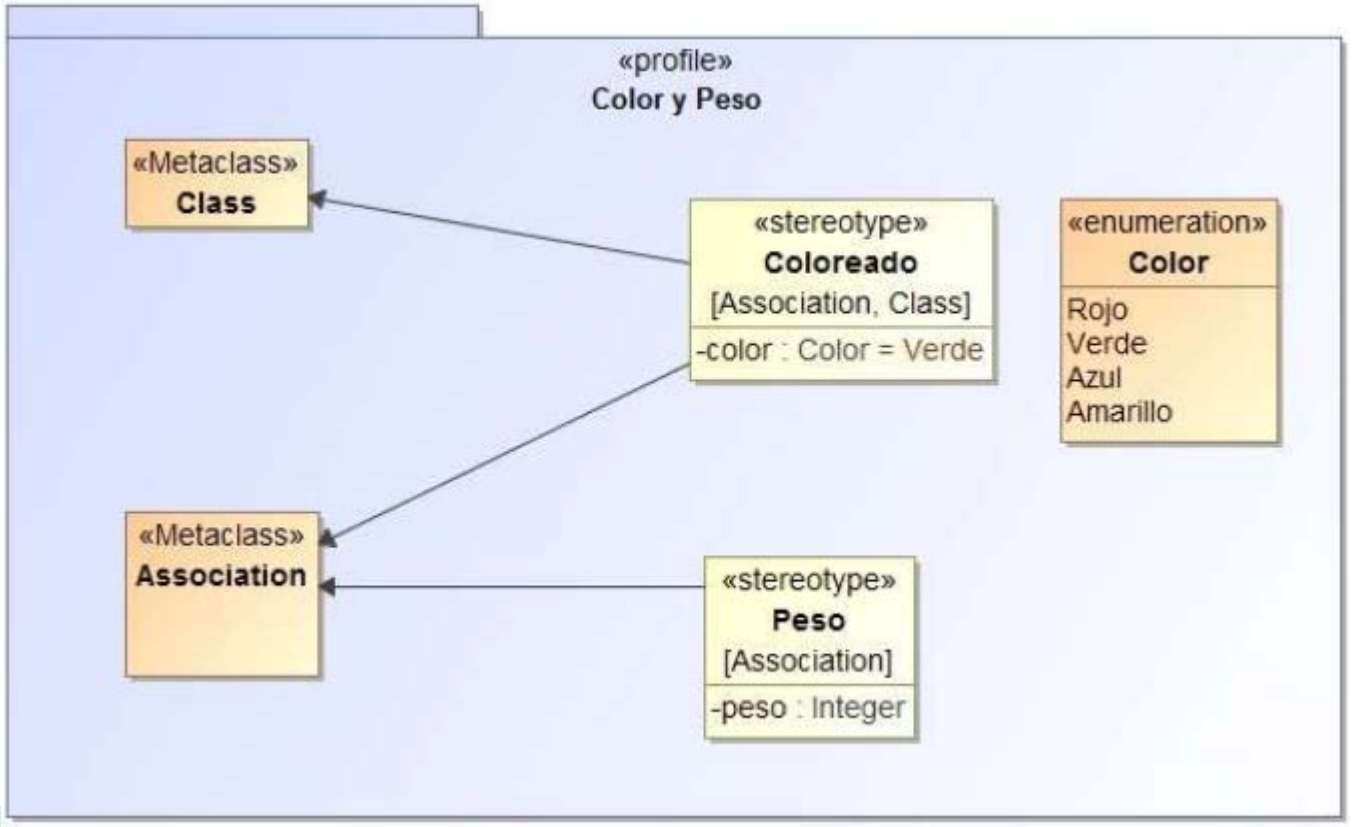
Definición de estereotipos

- En el nivel de metamodelado (M2), el estereotipo se asocia a los elementos del metamodelo que puede extender
  - Es un tipo especial de relación llamada **extensión**
- La clase estereotipada se etiqueta con el estereotipo estándar <<metaclass>> y el nuevo elemento, el estereotipo, se etiqueta con el estereotipo estándar <<stereotype>>

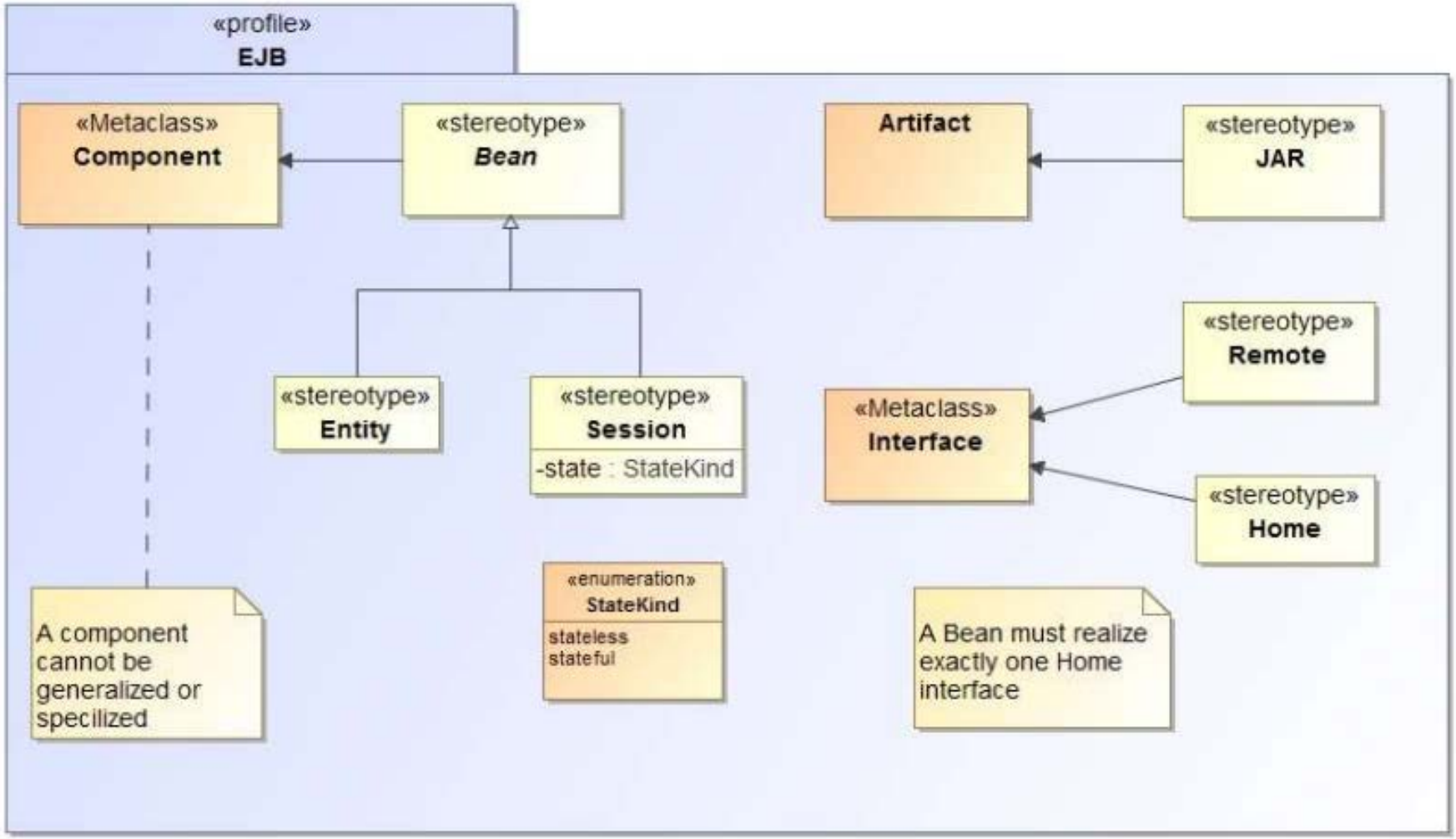


Definición de estereotipos: Ejemplo (I)

- Pequeño ejemplo de perfil UML, que va a definir dos nuevos elementos que pueden ser añadidos a cualquier modelo UML: colores y pesos.

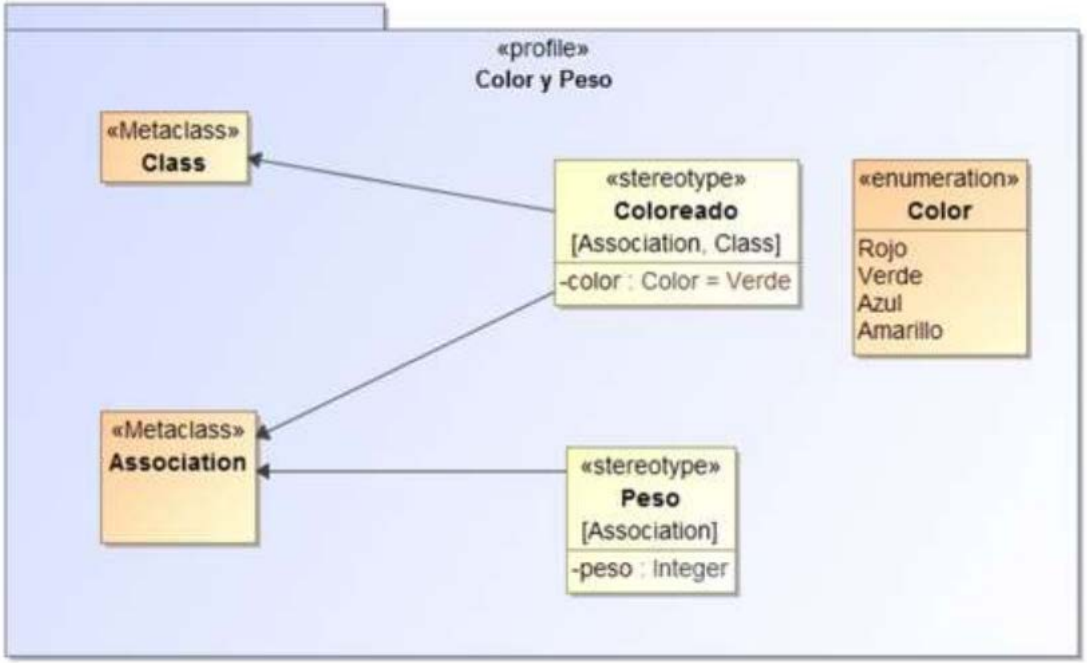


Ejemplo: EJB profile



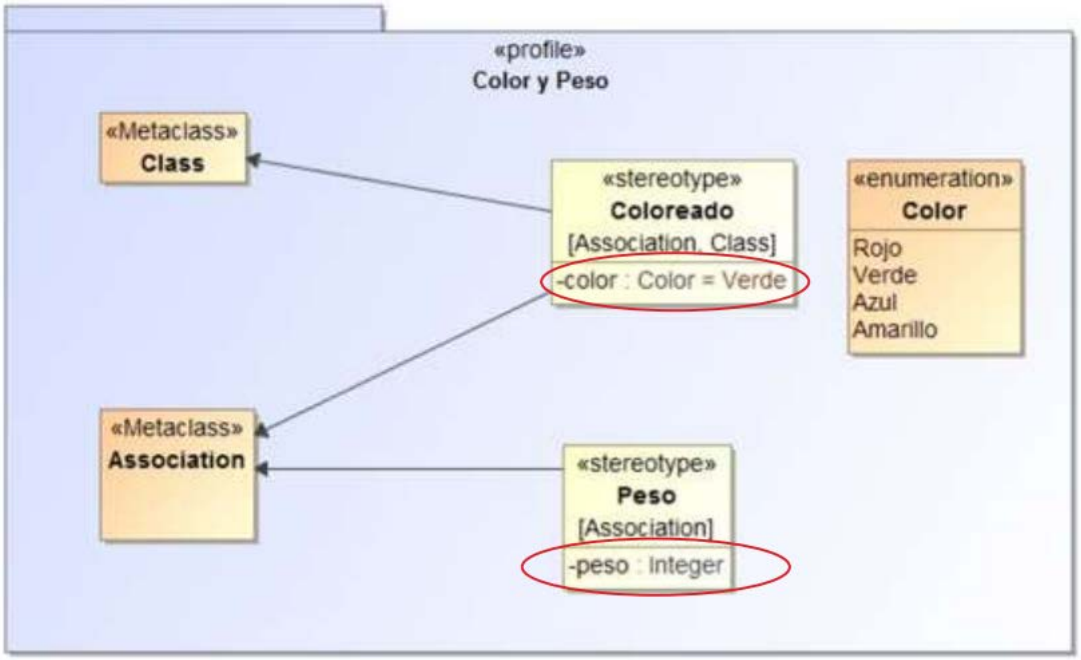
Definición de estereotipos: Ejemplo (II)

- El perfil define dos estereotipos, *Coloured* y *Weighed*, que proporcionan color y peso a un elemento UML
  - En el ejemplo, sólo las clases y asociaciones de UML pueden colorearse, y solo las asociaciones pueden tener un peso



Definición de etiqueta

- Define un atributo adicional que se asocia a una metaclase del metamodelo extendido por un perfil
- Ha de contar con un nombre y un tipo, y se define dentro de un determinado estereotipo

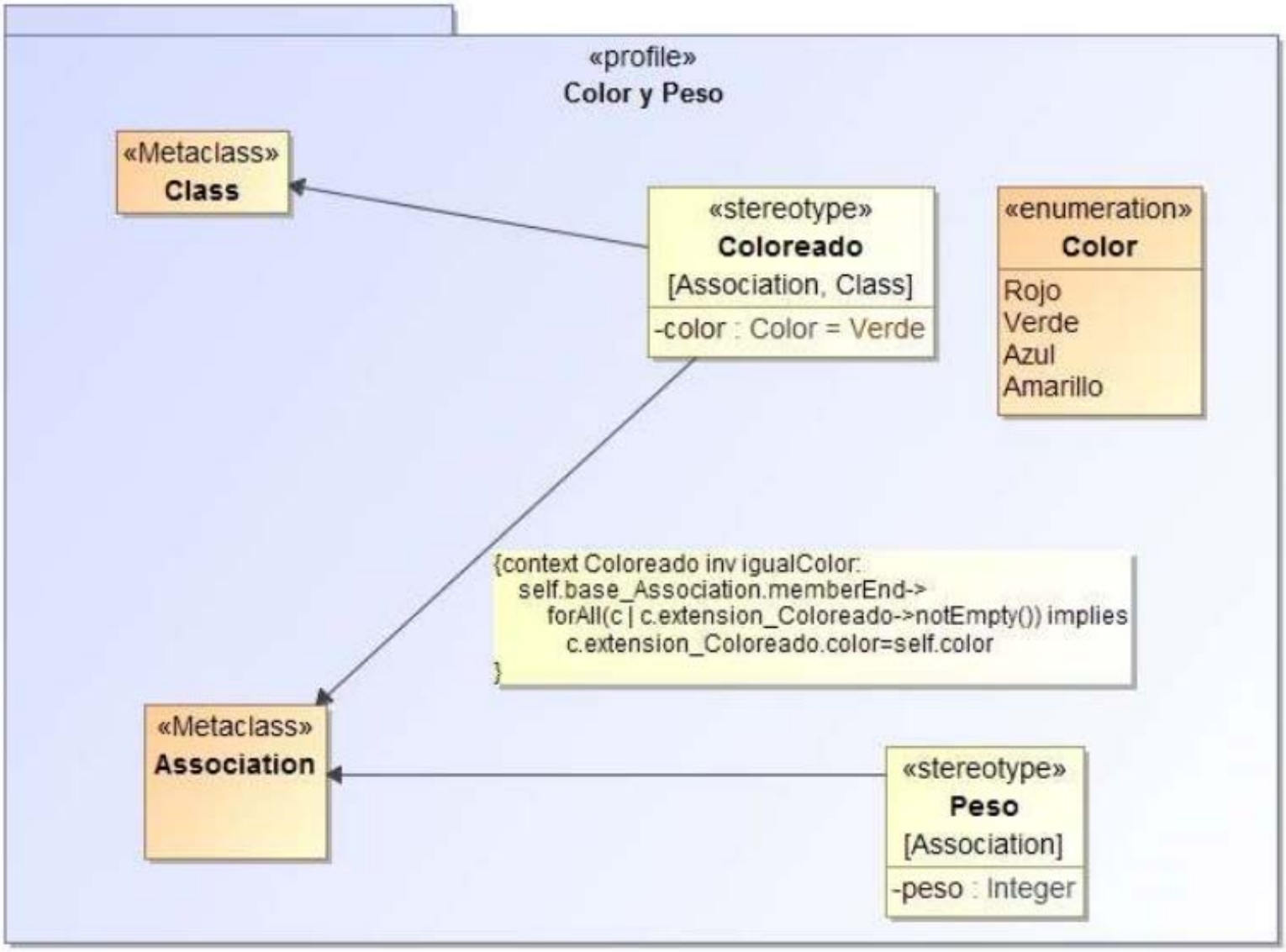


Restricciones

- A los estereotipos es posible asociarles restricciones, que imponen condiciones sobre los elementos del metamodelo que han sido estereotipados
  - Por ejemplo, si dos o más clases están unidas por una asociación coloreada, el color de las clases debe coincidir con el de la asociación
  - Puede expresarse en lenguaje natural o en OCL (Objcect Constrain Languaje), un lenguaje funcional asociado a UML

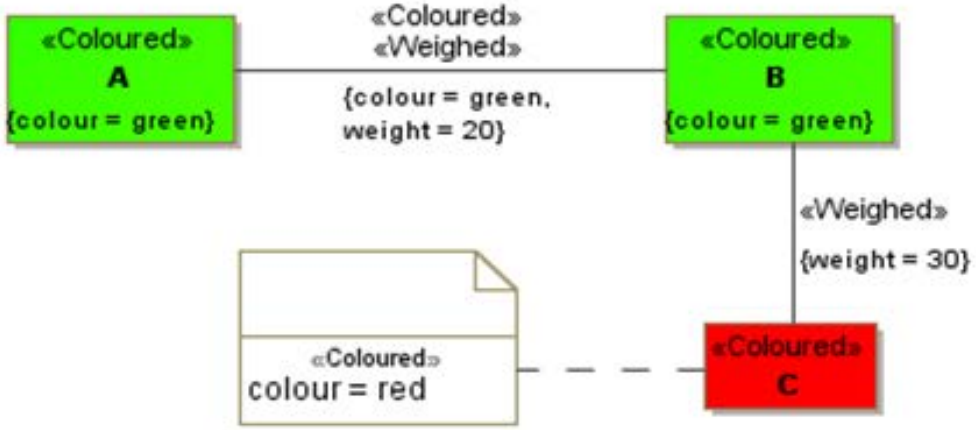
```
context Coloured inv sameColour:
    self.base_Association.memberEnd->
        forAll(c | c.extension_Coloured->notEmpty()) implies
            c.extension_Coloured.colour=self.colour
```

Restricciones

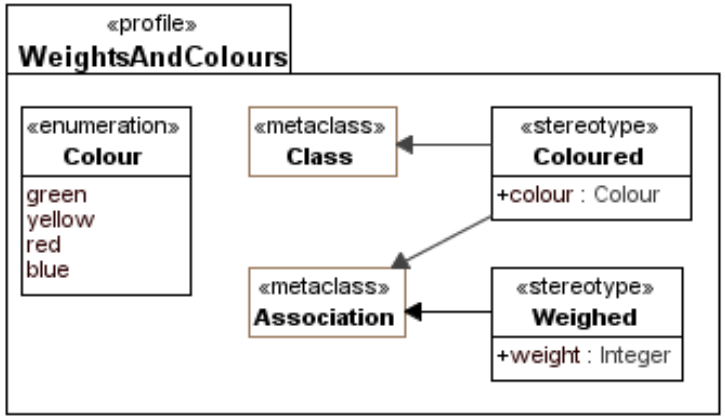


Ejemplo

- Ejemplo de uso del perfil UML con colores y pesos



- Las definiciones de etiqueta aparecen de diferentes formas



Máster en Ingeniería Web y Tecnologías RIA

UNIVERSIDAD DE MÁLAGA

MÓDULO 4

Ingeniería Web. Modelado

Antonio Vallecillo
av@lcc.uma.es

Loli Burgueño
loli@lcc.uma.es

Nathalie Moreno
moreno@lcc.uma.es

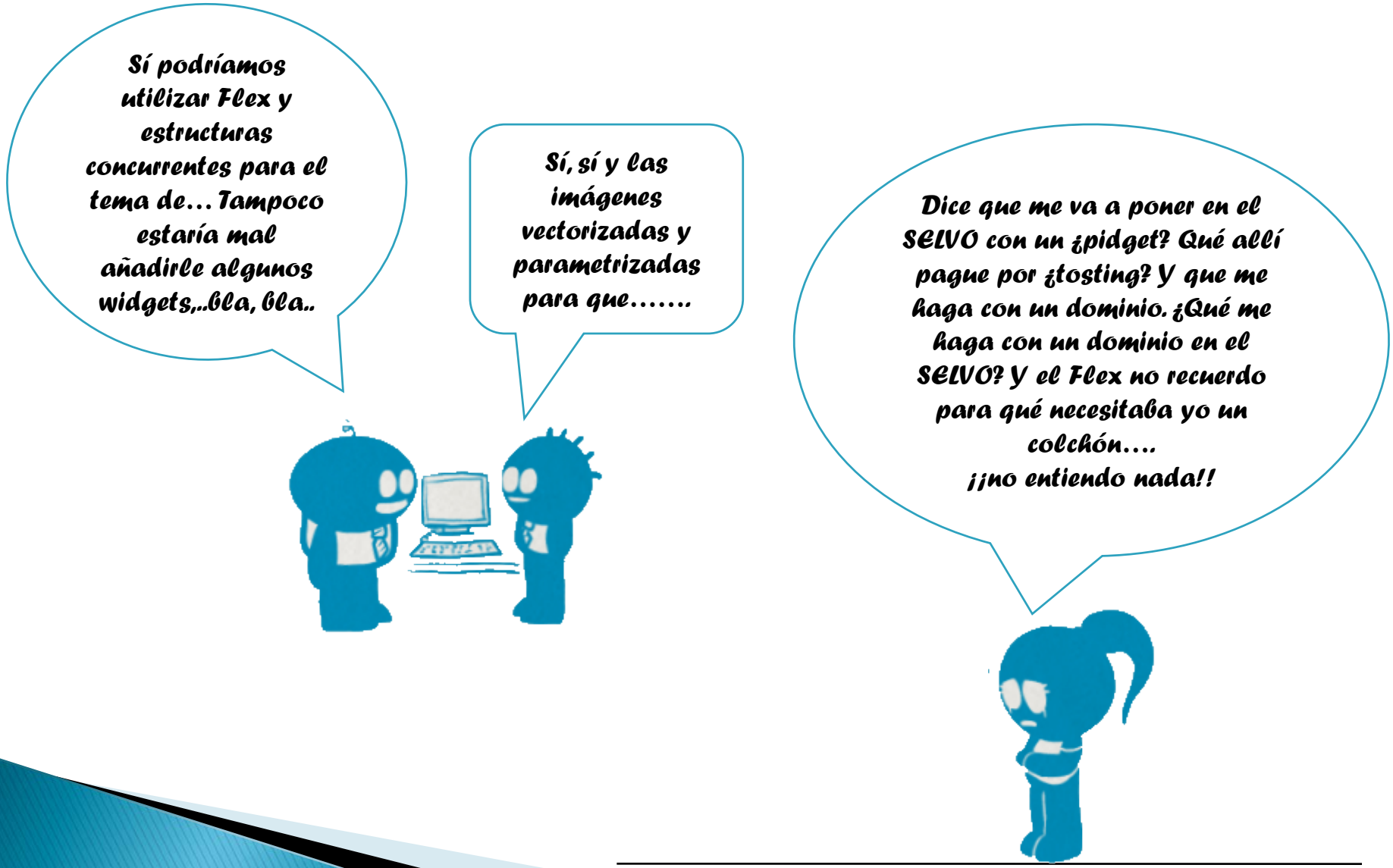
Índice

- BLOQUE I: Introducción
  - Necesidad de una Ingeniería Web
  - Estrategias de desarrollo de aplicaciones Web
  - Introducción a los Lenguajes de Modelado
- **BLOQUE II: Lenguajes de modelado**
  - Lenguajes de modelado de propósito general
    - UML y Perfiles UML
  - Lenguajes de modelado de propósito específico (DSL)
    - Definición de un DSL
    - DSLs basados en UML: UWE
    - DSLs basados en notaciones propias: WebML

La Torre de Babel (I)

La Torre de Babel (II)

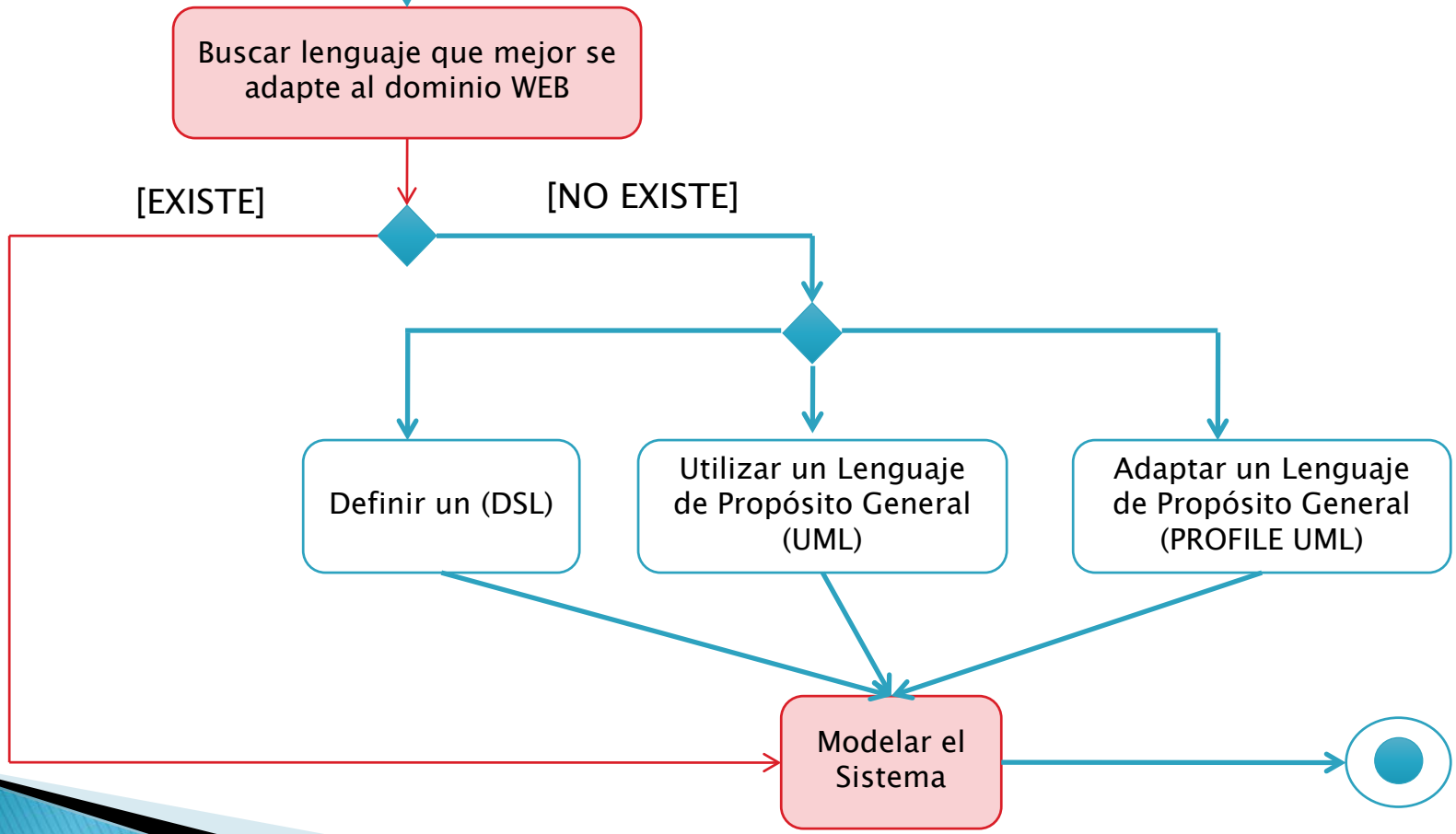
La Torre de Babel (III)



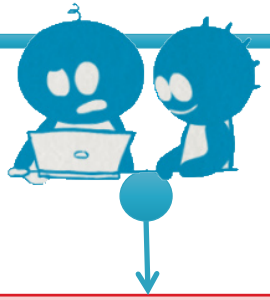
La Torre de Babel (IV)

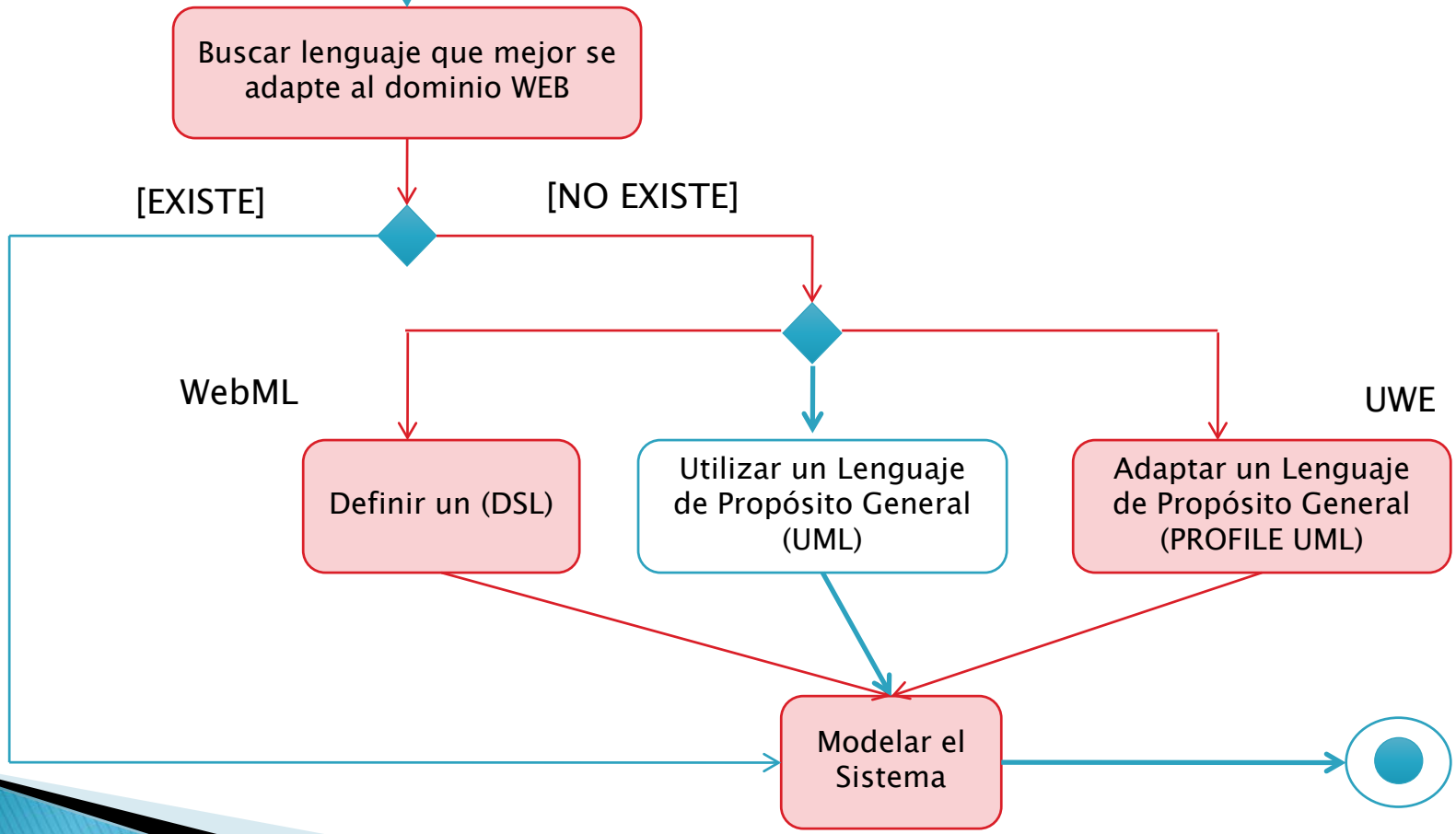
La Torre de Babel (V)

2010: Elección de un lenguaje de modelado



1990-1995: Elección de un lenguaje de modelado





Antes de continuar...

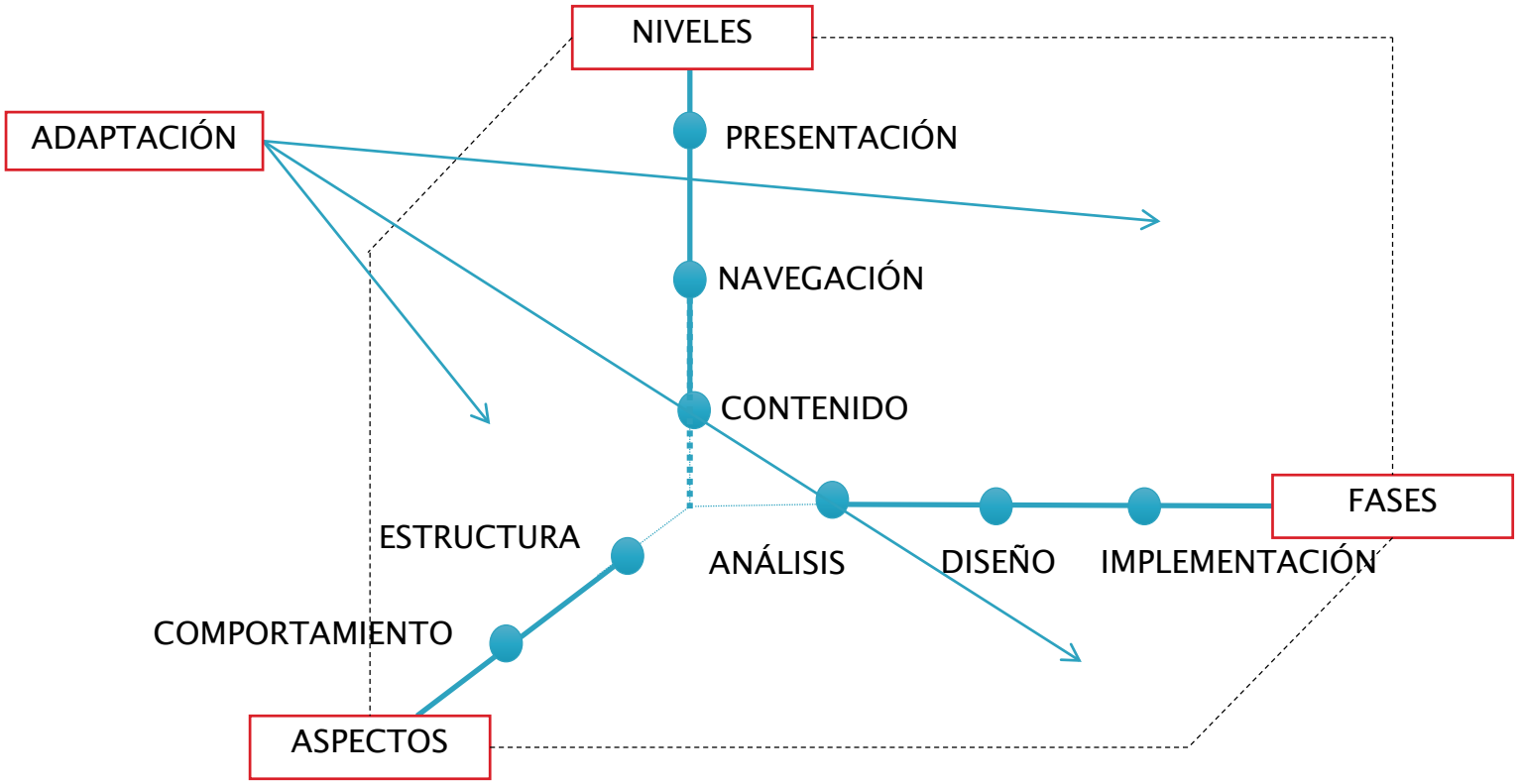
- Instalar el plugin de MagicUWE
  - Con permisos de administración, ejecutar consola
  - Java –jar MagicUWEv1.3.2.Installer.jar
- Instalar el profile de UWE
  - Descomprimir el fichero UWE_Profile_v1.8.zip en el directorio profiles de MagicDraw
- Desde MagicDraw
  - File > New Project > Project from template >UWE
  - File > Use Module >UWE_Profile

DSLs para Ingeniería Web basados en UML: UWE

Visión genérica de UWE (I)

- La propuesta de UWE incluye:
  - Un lenguaje de modelado: representar los modelos de las aplicaciones Web
    - El lenguaje se define como una extensión de UML (profile)
  - Un proceso de desarrollo: guiar al usuario en el desarrollo de una aplicación Web
    - Dirigido por la información (“content first”)
    - Dirigido por la funcionalidad (“business process first”)
    - Dirigido por la interfaz
  - Herramientas de modelado y generación semi-automática de código.
    - Modelado => MagicUWE
    - Generación de código =>UWE4JSF

Dimensiones del modelado Web con UWE



En la fase de Análisis

- Los requisitos funcionales son especificados utilizando:
  - 1.- Casos de Uso: Diagramas de casos de uso UML estereotipados
    Workflows: Diagramas de actividad UML
- Los requisitos de datos son especificados utilizando:
  Modelos de Dominio: Diagramas de clases UML

UWE: Modelos y su representación

Fase de diseño

- **En la fase de Diseño**
  - Aspectos de información
    - 2.- Modelo de contenido: Diagramas de clases UML
  - Estructura de navegación
    - 3.- Modelo de navegación: Diagrama de clases UML estereotipados
  - Funcionalidad
    - 4.- Modelo de procesos
      - 4.1.- Modelo de clases de procesos: Diagramas de clases UML estereotipados
      - 4.2.- Modelo de flujo de procesos :Diagramas de actividad UML estereotipados
  - Aspectos relativos a la presentación
    - 5.- Modelo de presentación: Diagramas de estructuras compuestas UML estereotipados

UWE: Modelos y su representación

Fases de adaptación e implementación

- **En la fase de Adaptación**

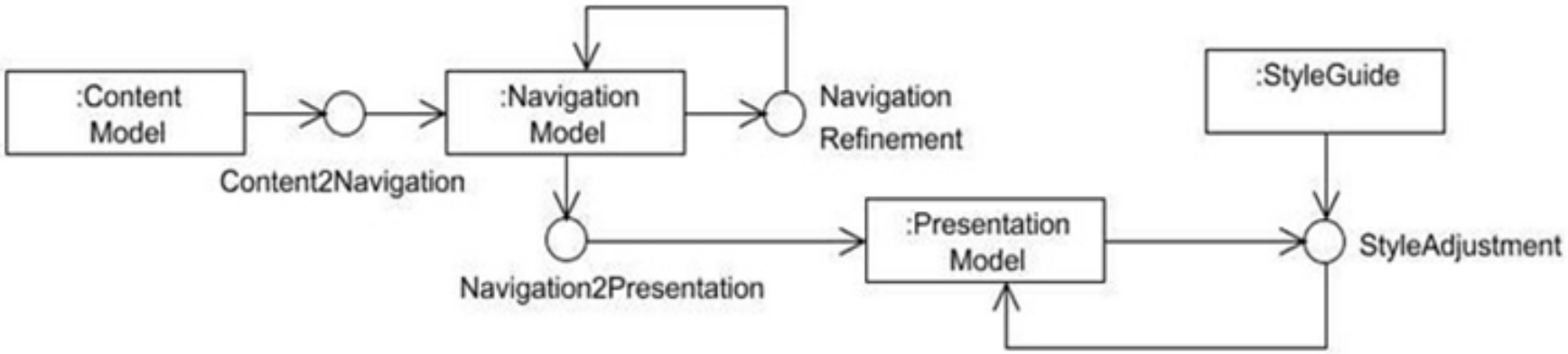
  6.- Modelo de adaptación: Paquetes UML estereotipados

- **En la fase de Implementación**
  - Se genera el código de la aplicación
  - Actualmente en la plataforma JSF

Proceso de desarrollo con UWE

UWE: Proceso

- Proceso **iterativo** e **incremental**



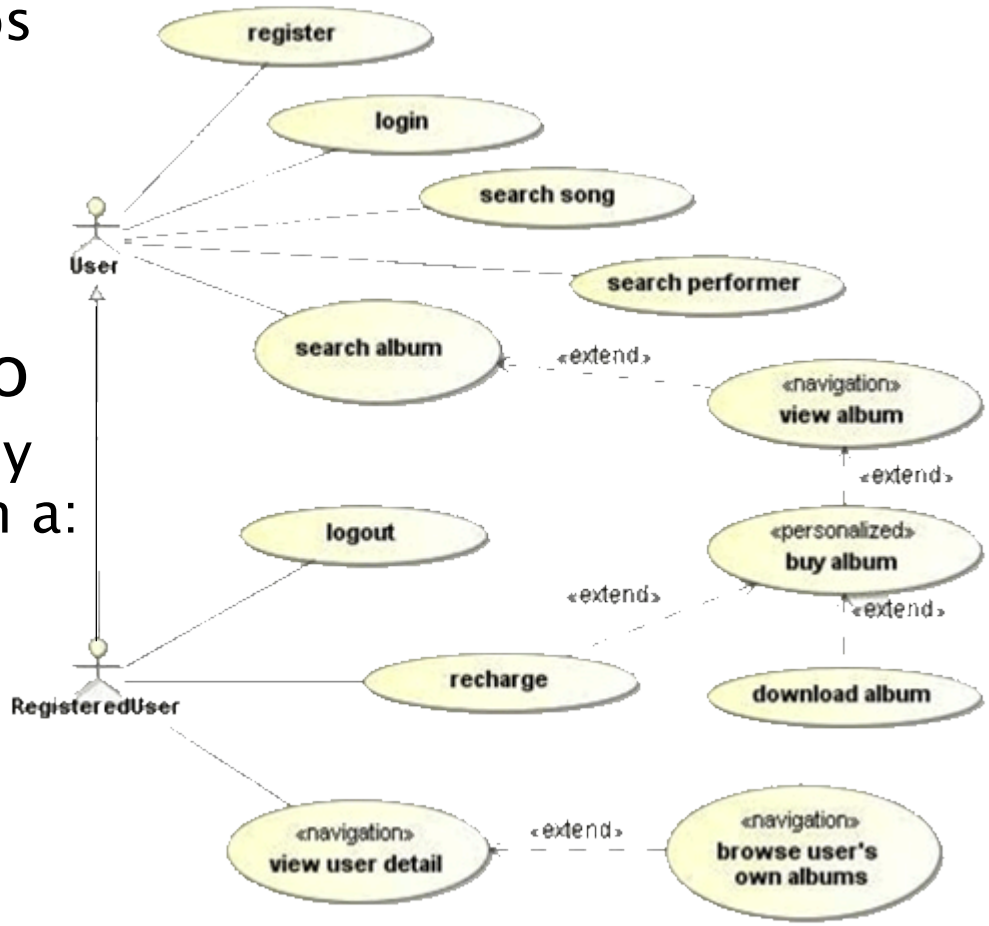
Modelos y Elementos de Modelado

Ejemplo: www.mp3.com

- Site
  - Contiene información sobre álbumes, canciones y artistas
  - Esta información es de libre disposición para todos los usuarios
    - Sólo los usuarios registrados pueden bajarse un álbum
    - Para poder bajárselo es necesario tener una cuenta de crédito de prepago (ej. Paypal)
    - Las cuentas pueden recargarse

Análisis: Modelado de requisitos

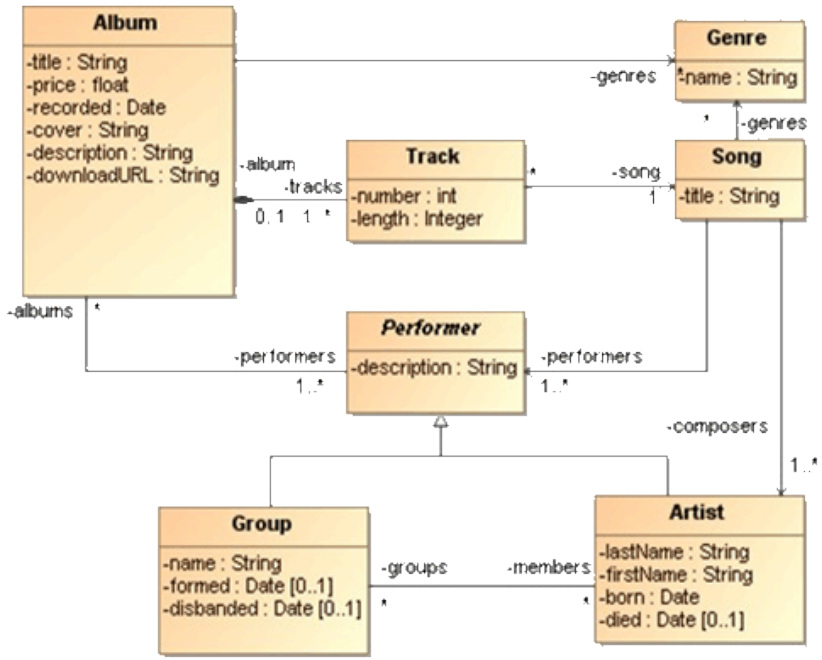
- Objetivo
  - Visualizar gráficamente los requisitos del proyecto.
    - Funcionales
    - Navegacionales
    - De adaptación
- Elementos de modelado
  - Analizar los casos de uso y determinar cuáles refieren a:
    a) Aspectos relativos a procesamiento <<webProcess>>
    b) Aspectos de la navegación <<navigation>>
    c) Requieren adaptación <<personalized>>



Diseño: Modelo de contenido

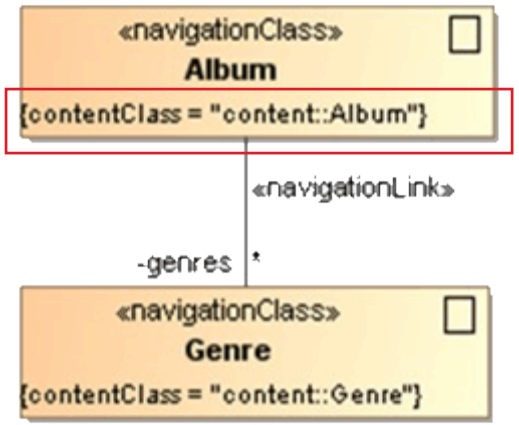
- Objetivo
  - Representar el dominio de la información persistente.
  - Información necesaria para el procesamiento de la lógica del negocio
- Elementos de modelado
  - Diagrama de clases UML plano, sin semántica adicional.
    - Clases
    - Asociaciones
    - Agregaciones/Composición
    - Jerarquías
    - …

- Ejemplo



Diseño: Modelo de navegación

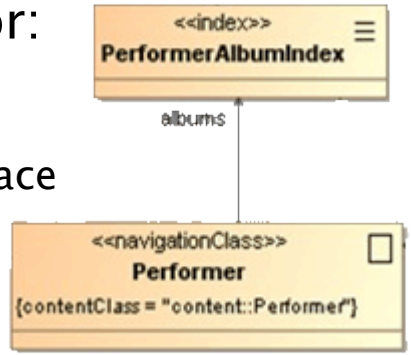
- Objetivo
  - Representar los nodos y enlaces de la estructura de hipertexto.
    - **Nodo**: Clase visitada por el usuario/sistema utilizando el navegador.
    - **Enlace**: Estructura usada para acceder desde un nodo fuente a un nodo destino.
  - Diseñar los caminos navegacionales
- Elementos de modelado
  - Estructura de hipertexto:
    - Nodos: <<navigationClass>>
    - Enlaces: <<navigationLink>>
    - Primitivas de acceso: <<index>>
      <<menu>>
      <<guidedTour>>
      <<query>>



Diseño: Modelo de navegación

Primitiva <<index>>

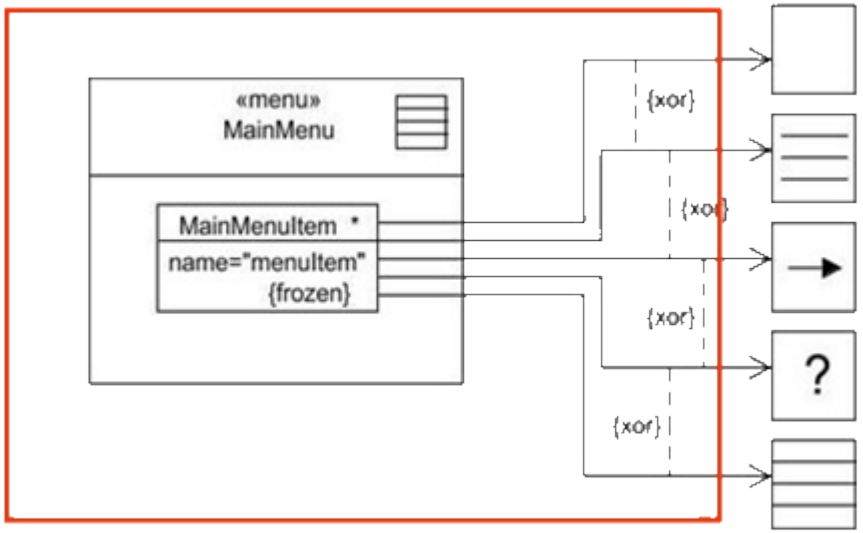
- Permite seleccionar una instancia de una de las clases del modelo de contenido
- Agrupa en un nodo un conjunto items
  - Cada item es un objeto con un nombre y un enlace a una instancia de una clase navegacional
  - Todos los enlaces apuntan a la misma clase navegacional
- Partiendo de esa lista de enlaces, el usuario puede seleccionar un elemento en particular para continuar la navegación
- El conjunto de instancias de ente las cuales se permite al usuario realizar su elección viene determinado por el enlace entrante y su predecesor:
  - *Query*
  - Clase navegacional
    - necesario especificar el atributo del que proviene en el enlace
    - e.g. albums
  - Menu



Diseño: Modelo de navegación

Primitiva <<menu>>

- Agrupa en un nodo un conjunto de enlaces de salida a distintas clases navegacionales heterogéneas (índices, visitas guiadas, consultas, otros menús,…)
- Estas clases navegacionales destino son disjuntas entre sí
- Nota: un menú en el modelo de navegación no siempre se corresponde con un menú en la interfaz de usuario



Diseño: Modelo de navegación

Nodos de decisión

- Primitiva <<guidedTour>> 

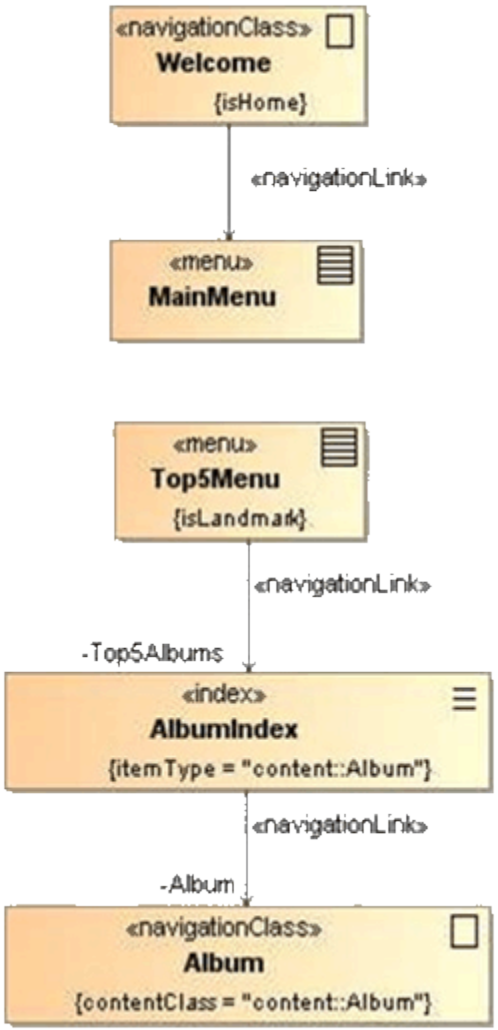
  - Recibe como entrada un conjunto de instancias
  - Permite el acceso secuencial a las instancias (navegar hacia adelante y hacia atrás)
    - El orden es especificado usando una FilterExpression
  - La instancia seleccionada en un momento determinado es la entrada a una clase navigacional

- Primitiva <<Query>> 

  - Establece la posibilidad de buscar instancias de un nodo destino, las cuales satisfacen una condición o filtro establecido en la propiedad FilterExpression

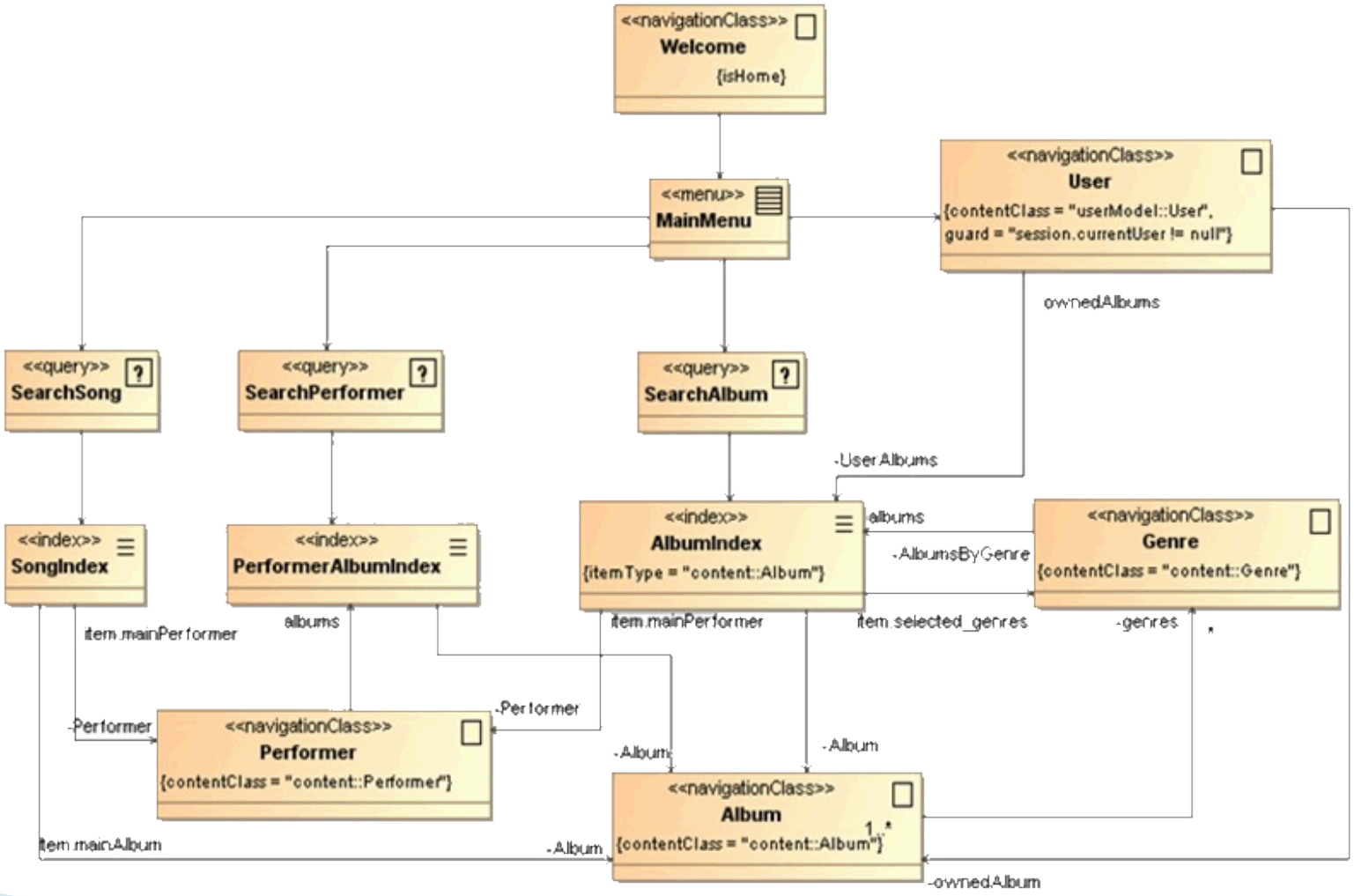
Diseño: Modelo de navegación
Valores etiquetados

- {isHome} indica el nodo inicial de la aplicación
  - Dicho nodo no puede contener enlaces de entrada
- {isLandmark} indica que el nodo es alcanzable desde cualquier lugar de la aplicación (desde cualquier otro nodo del diagrama de navegación)



Diseño: Modelo de navegación

Ejemplo



Diseño: Modelo de navegación
Vistas

- Para un mismo modelo de contenido se pueden considerar varias vistas del modelo navigacional (que se reflejan en varios diagramas, uno por vista):
  - Si el número de nodos de navegación es grande =>construir vistas parciales
  - Si la aplicación va a ser utilizada por diferentes usuarios con distintos privilegios sobre ella =>construir una vista por cada usuario
  - Si la aplicación va a ser ejecutada en diferentes entornos => construir una vista por cada entorno

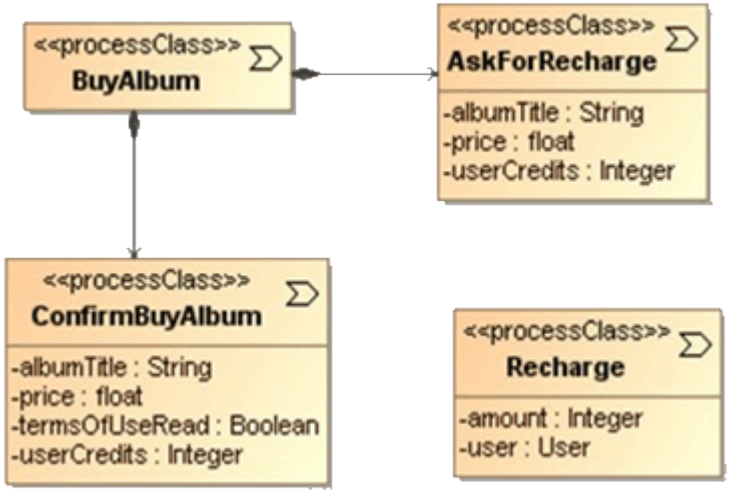
Diseño: Modelo de procesos

- Objetivo
  - Representar los **aspectos dinámicos** de una aplicación Web.
  - Especificar la funcionalidad: transacciones, flujos de trabajo
- Elementos de modelado
  - Diagrama con las clases de procesos
    - Diagrama de clases UML => Se definen las “clases procesos” así como sus relaciones
    - Se parte del diagrama de casos de uso
    - Se identifican los casos de uso no navegacionales (**<<webProcess>>**)
    - Se define un diagrama de clases UML representando las clases proceso.
  - Diagrama de integración: Se integran las “clases procesos” en el modelo de navegación
  - Diagramas de actividad UML: Se define el comportamiento a través de flujos de procesos.

Diseño: Modelo de procesos

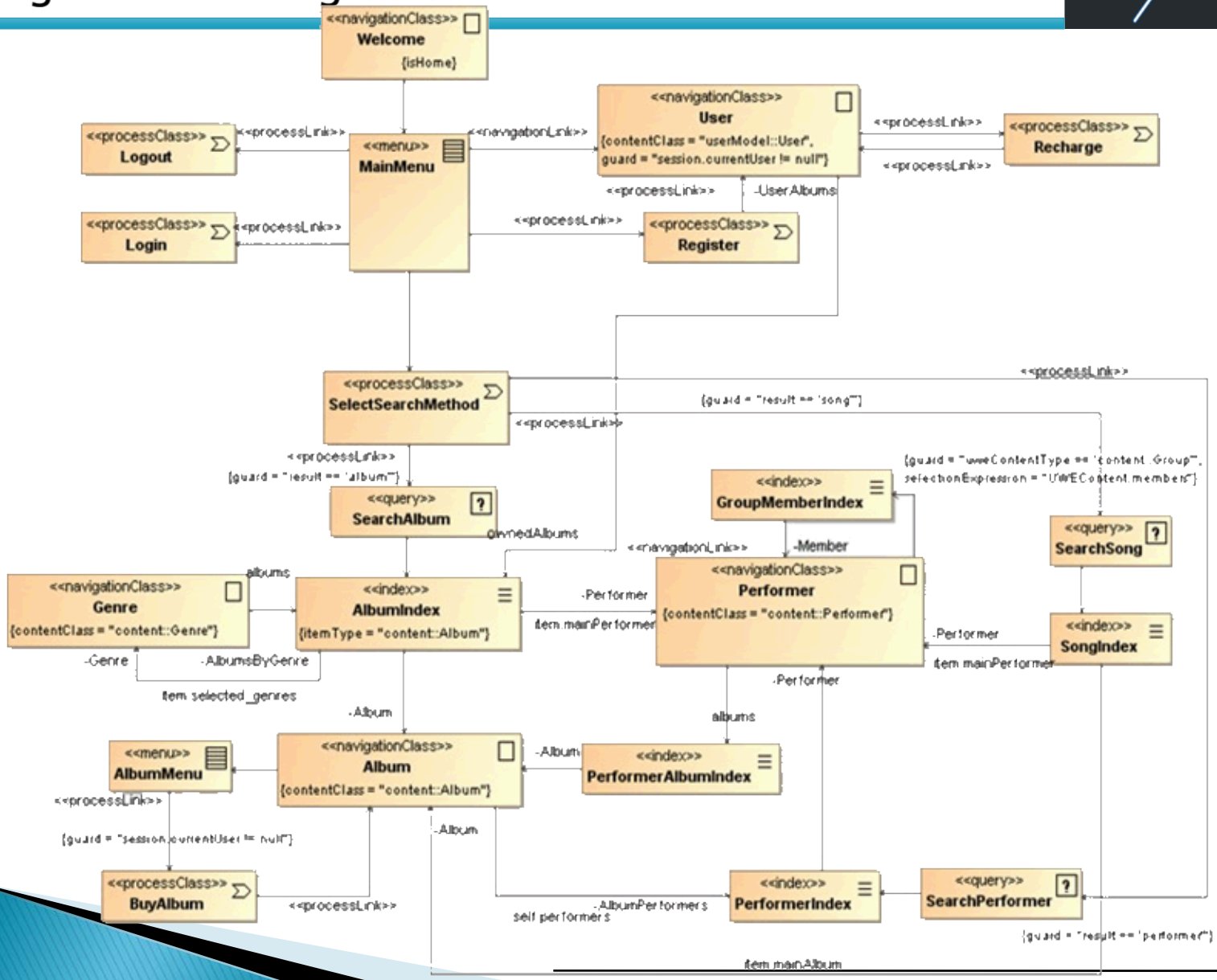
Diagrama de clases de procesos

- Diagrama de clases procesos
  - <<process class>> representa una actividad ejecutada por el usuario o sistema dentro de la aplicación Web.
  - <<process link>> indica un punto de entrada o salida a un proceso dentro de la estructura de navegación.



Diseño: Modelo de procesos

Diagrama de integración



Diseño: Modelo de procesos

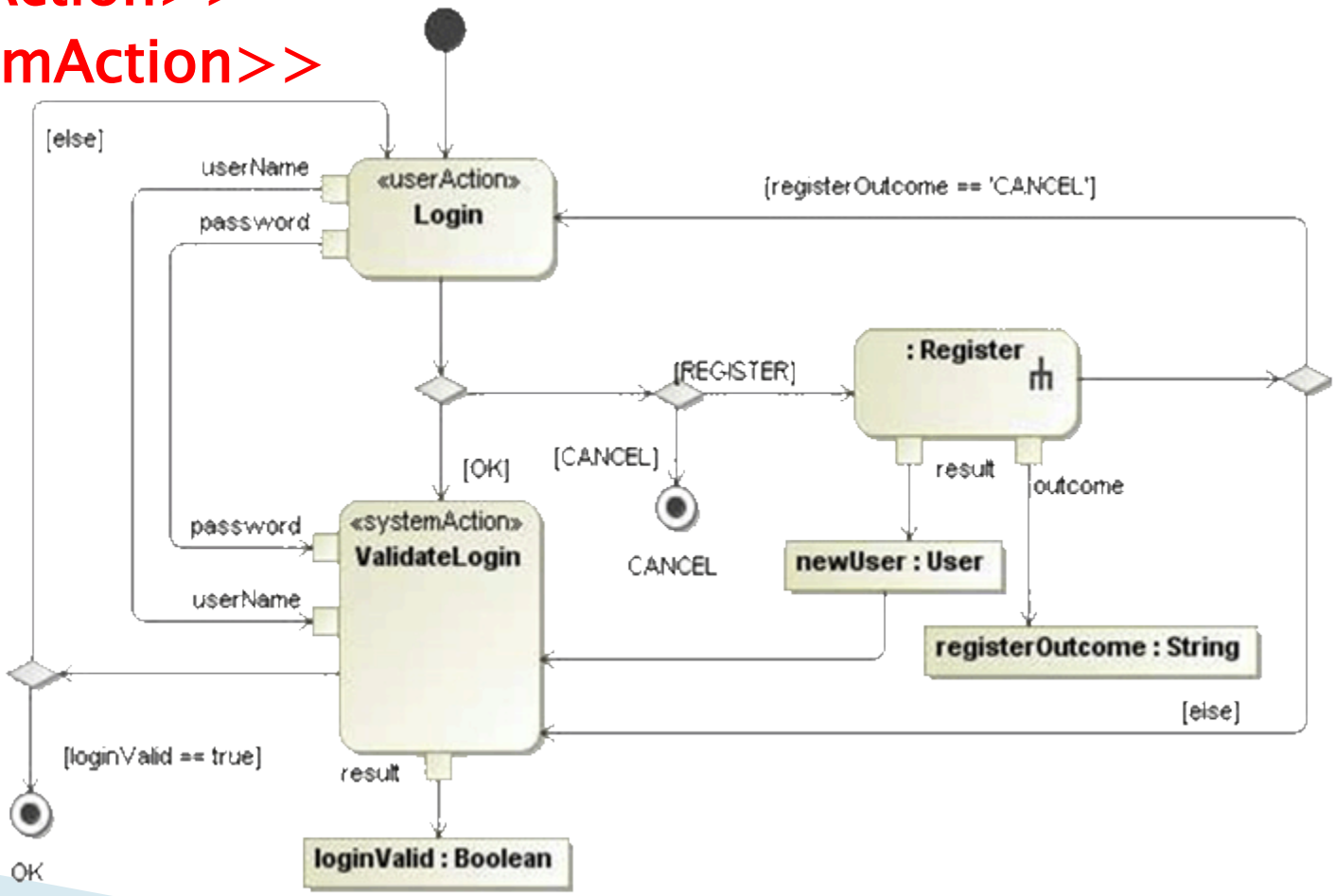
Diagrama de flujo de procesos

- Los modelos de flujo de procesos describen el comportamiento de una aplicación Web
  - Representados mediante diagramas de actividad
  - Resultado de un refinamiento de los diagramas de actividad definidos en la fase de análisis de requisitos.
- Incluyen:
  - Un cjto de **acciones**
  - Un cjto de **elementos de decisión o nodos de control**
  - Un cjto de **instancias de clases que representan el flujo de información que se genera y consumen** los nodos de actividad

Diseño: Modelo de procesos

Diagrama de flujo de procesos

- Elementos de modelado:
  - <<userAction>>
  - <<systemAction>>



Diseño: Modelo de presentación

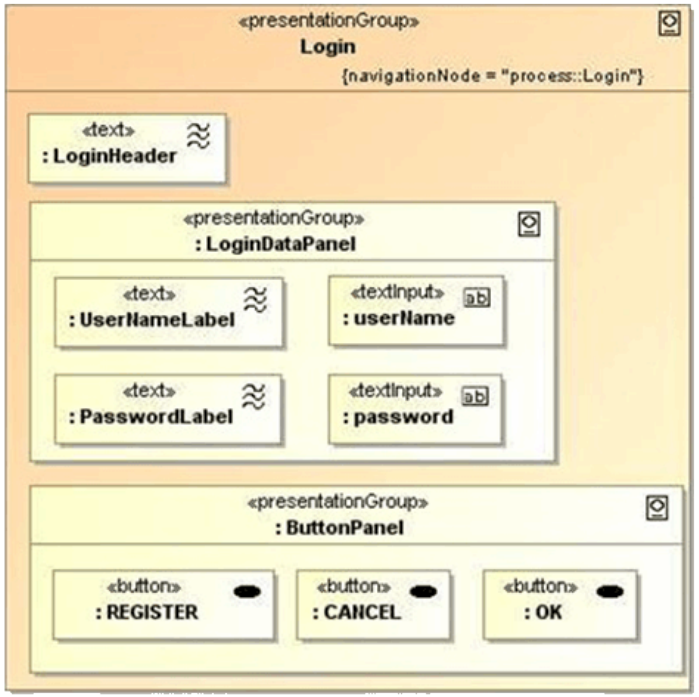
- Objetivo
  - Representar de forma abstracta los requisitos de presentación que subyacen bajo el modelo de navegación.
  - Características físicas como:
    - "color"
    - "posición"
    - "número de columnas de la tabla"
    - características específicas de una plataforma de implementación
    - …

    NO son tratadas en este modelo.
- Elementos de modelado
  - Estructura del modelo de presentación: Diagrama de estructuras compuestas de UML
  - Comportamiento del modelo de presentación: Diagramas de interacción (diagramas de secuencia)

Diseño: Modelo de presentación

Modelo de estructuras compuestas

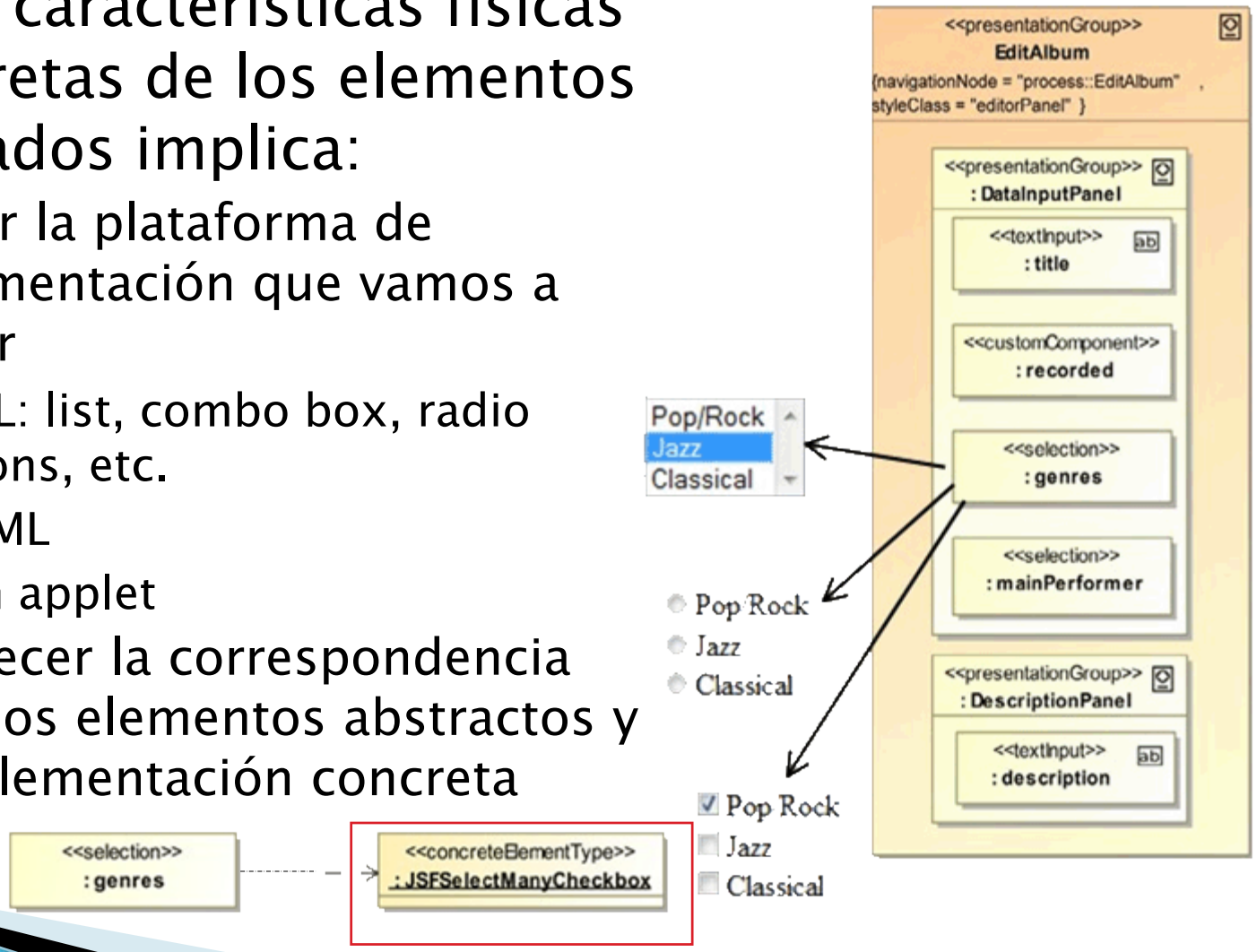
- La estructura del modelo de presentación se lleva a cabo con el diagrama de estructuras compuestas.
  - <<presentationGroup>> agrupa un conjunto de elementos de la interfaz de usuario que representan una unidad lógica de procesamiento.
  - <<page>> conjunto de elementos que serán presentados simultáneamente al usuario en respuesta a una petición del mismo.
- Elementos de la interfaz de usuario
  - <<anchor>>
  - <<text>>
  - <<image>>
  - <<textInput>>
  - <<choice>>
  - …



Diseño: Modelo de presentación

Modelo de estructuras compuestas

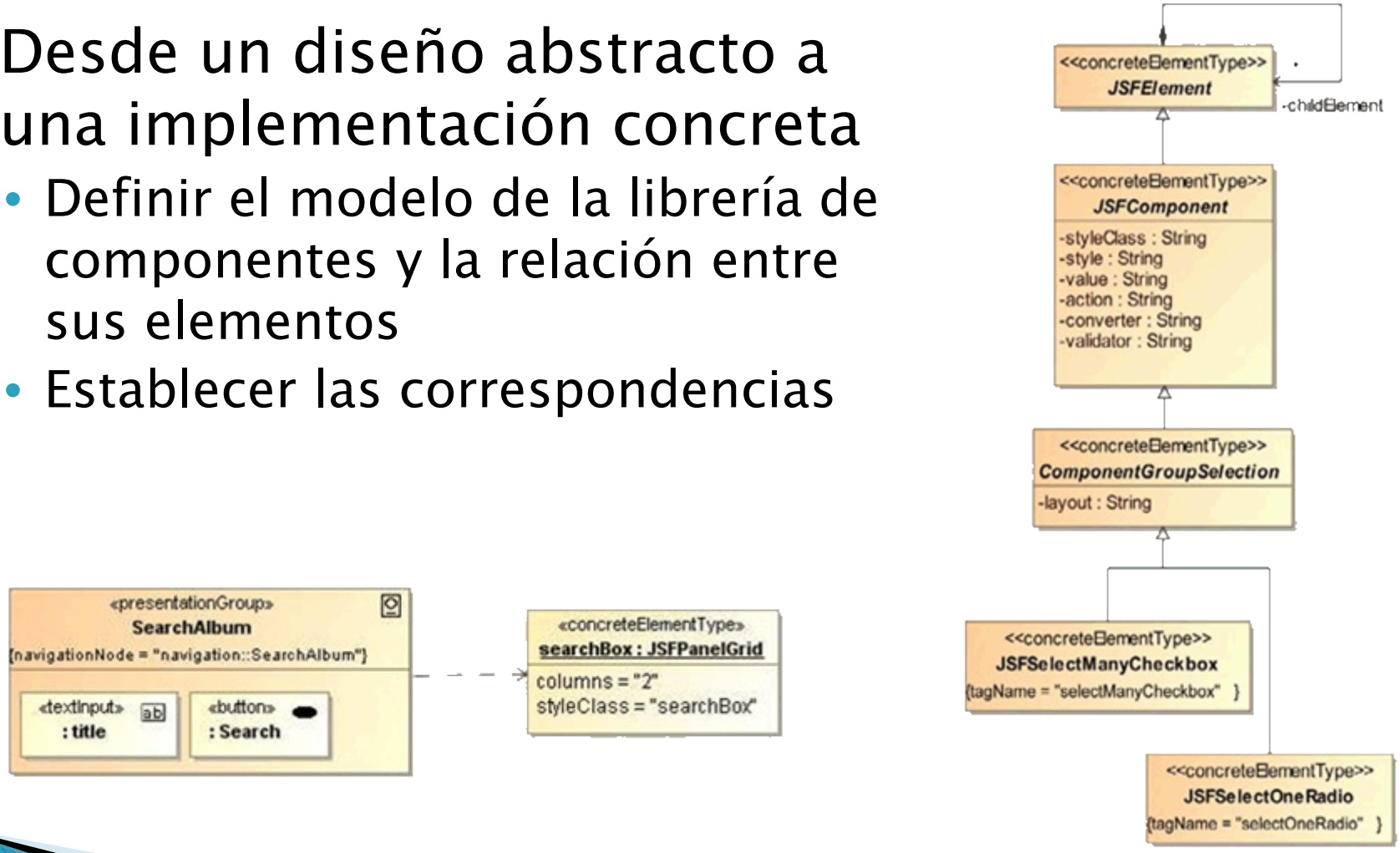
- Definir características físicas y concretas de los elementos modelados implica:
  - Decidir la plataforma de implementación que vamos a utilizar
    - HTML: list, combo box, radio buttons, etc.
    - DHTML
    - Flash applet
  - Establecer la correspondencia entre los elementos abstractos y la implementación concreta



Diseño: Modelo de presentación

Modelo de estructuras compuestas

- Desde un diseño abstracto a una implementación concreta
  - Definir el modelo de la librería de componentes y la relación entre sus elementos
  - Establecer las correspondencias



MagicUWE + JSF

- Fase de Modelado
  - UWE Profile (adaptado),
  - Common JSF component libraries

- Incluir en la carpeta profiles de MagicDraw
- File -> Use Module ->…

UWE: Aplicaciones Web Adaptativas

- UWE usa, para el modelado de aplicaciones Web adaptativas, una técnica basada en el modelado de aspectos.

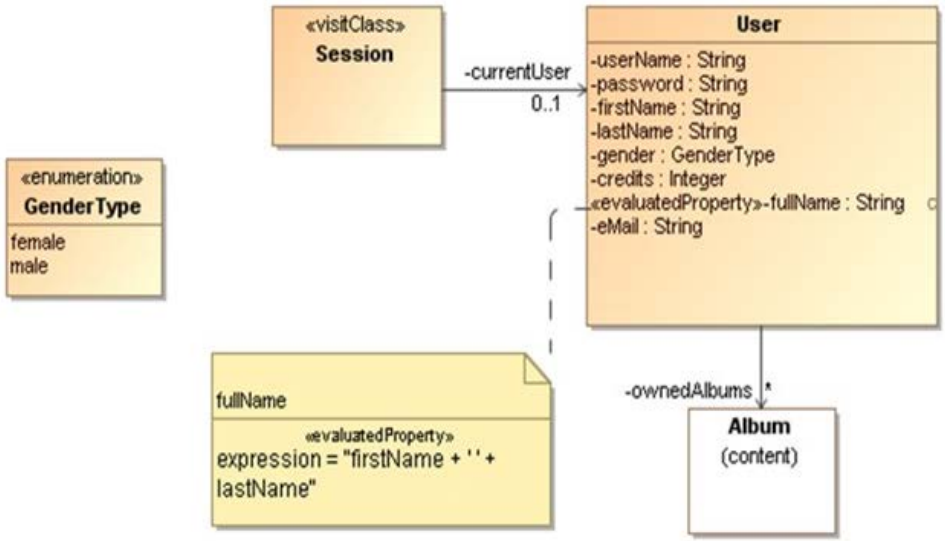
- Aplicaciones Web Adaptativas
  - Características del usuario: intereses, preferencias, conocimiento,…
  - Propiedades del contexto: localización (lugar y tiempo) y plataforma (HW, SW, red,..)

UWE: Técnicas de adaptación

- Adaptación del modelo de contenido
  - Determinar qué aspectos de la persistencia pueden variar
  - Insertar/Eliminar contenidos
- Adaptación del modelo de navegación
  - Ordenación de los enlaces
  - Generación/Ocultación de enlaces navegacionales
  - Anotación de enlaces.
- Adaptación del modelo de presentación.
  - Selección del lenguaje
  - Modificación de características (tamaño de fuentes, imágenes, cambios en el color de los elementos,…)

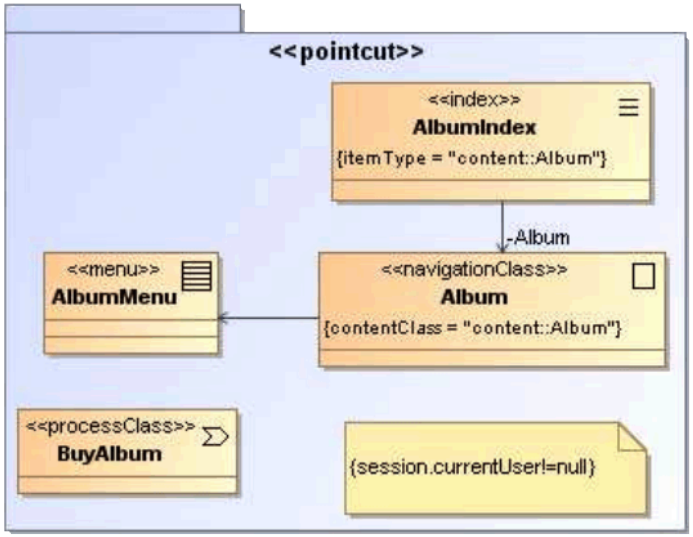
UWE: Modelo de usuario

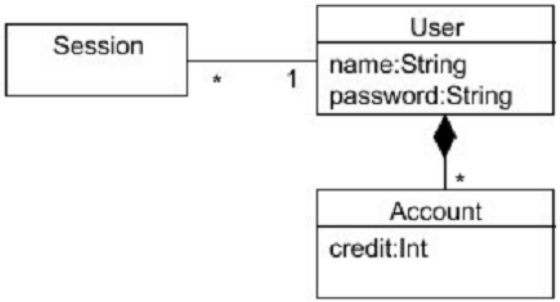
- Objetivo
  - Representación de la Sesión en curso del usuario y de información específica para el mismo.
- Elementos de modelado
  - Representado como un diagrama de clases UML al que se incorpora el estereotipo (según corresponda)
    - <<visitClass>> => Objeto representando la sesión del usuario.



UWE: Modelado Orientado a Aspectos

- Se identifican y definen:
  - Puntos de Corte: localizaciones en un modelo/programa donde se incorporarán aspectos adicionales a dicho modelo/programa
  - Avisos: Características que deben ser añadidas o ejecutadas en un punto de unión

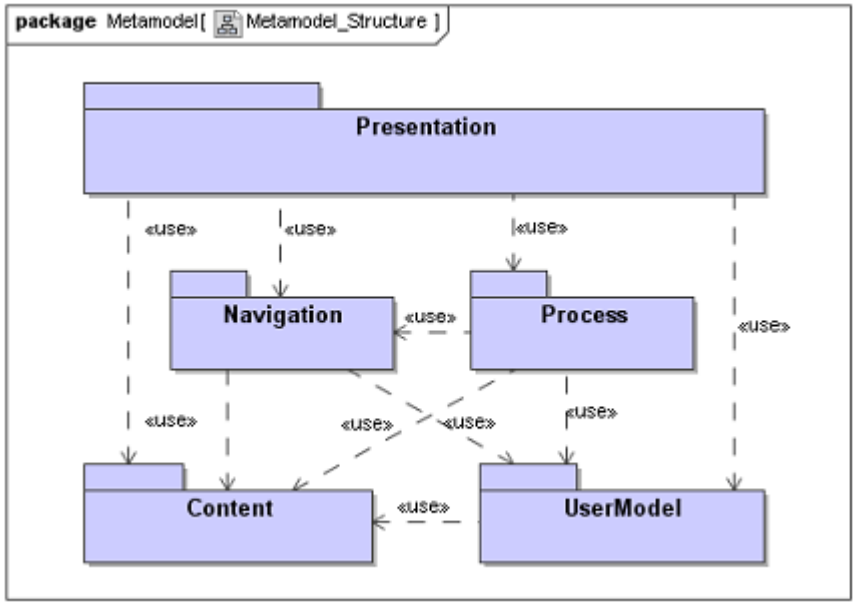


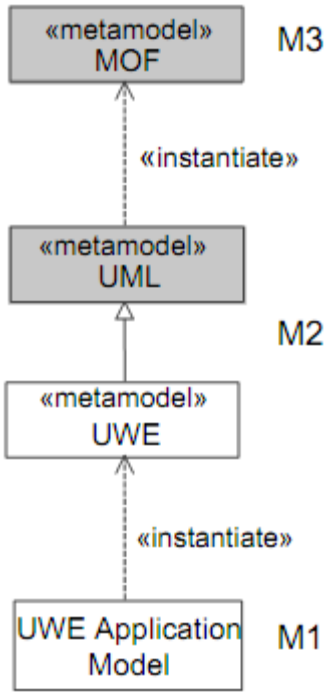


Metamodelos UWE

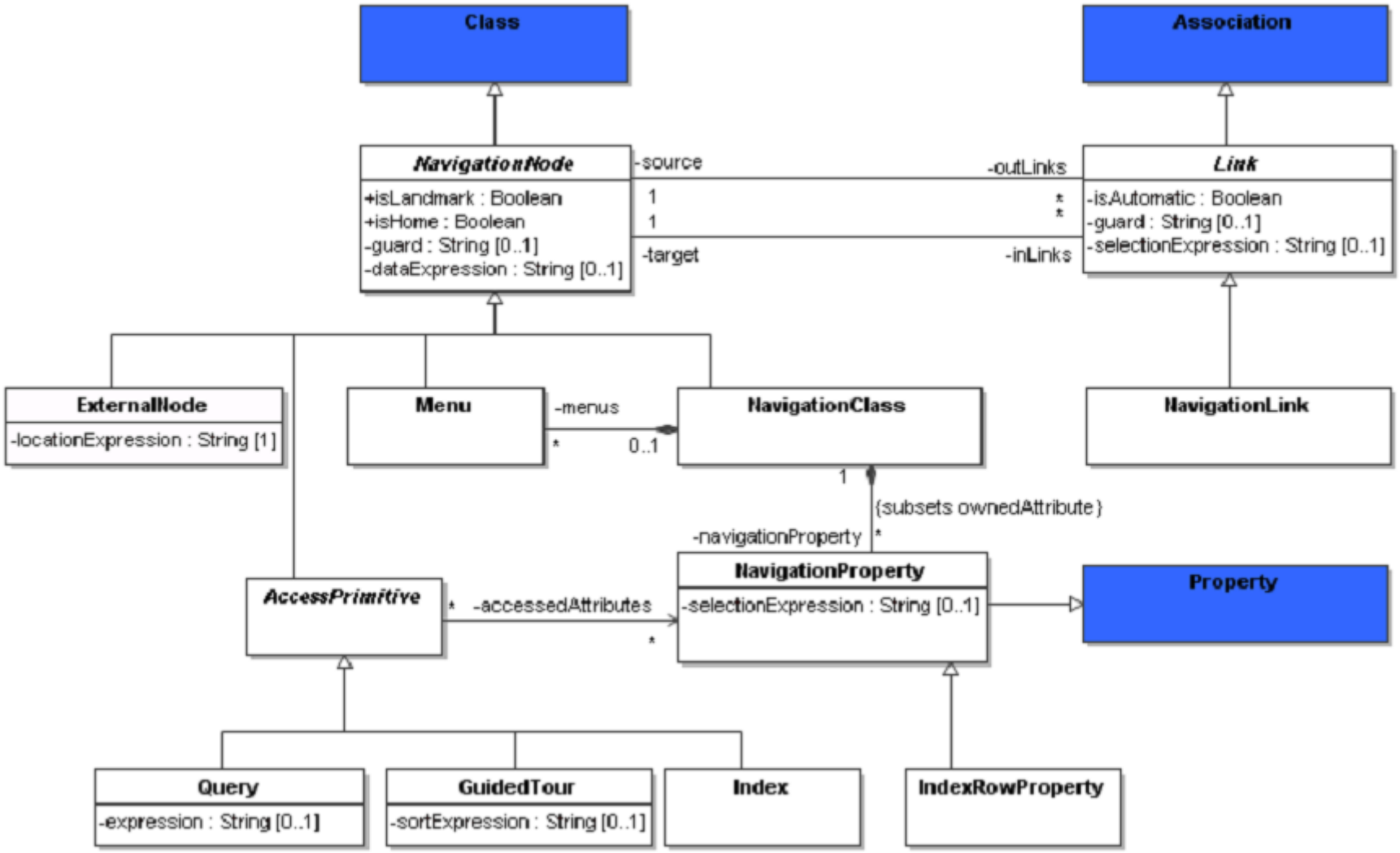
Metamodelo

- Definido como una extensión conservativa de UML
- Compatible con la arquitectura propuesta por OMG y sus estándares: OCL, XMI, MOF,.

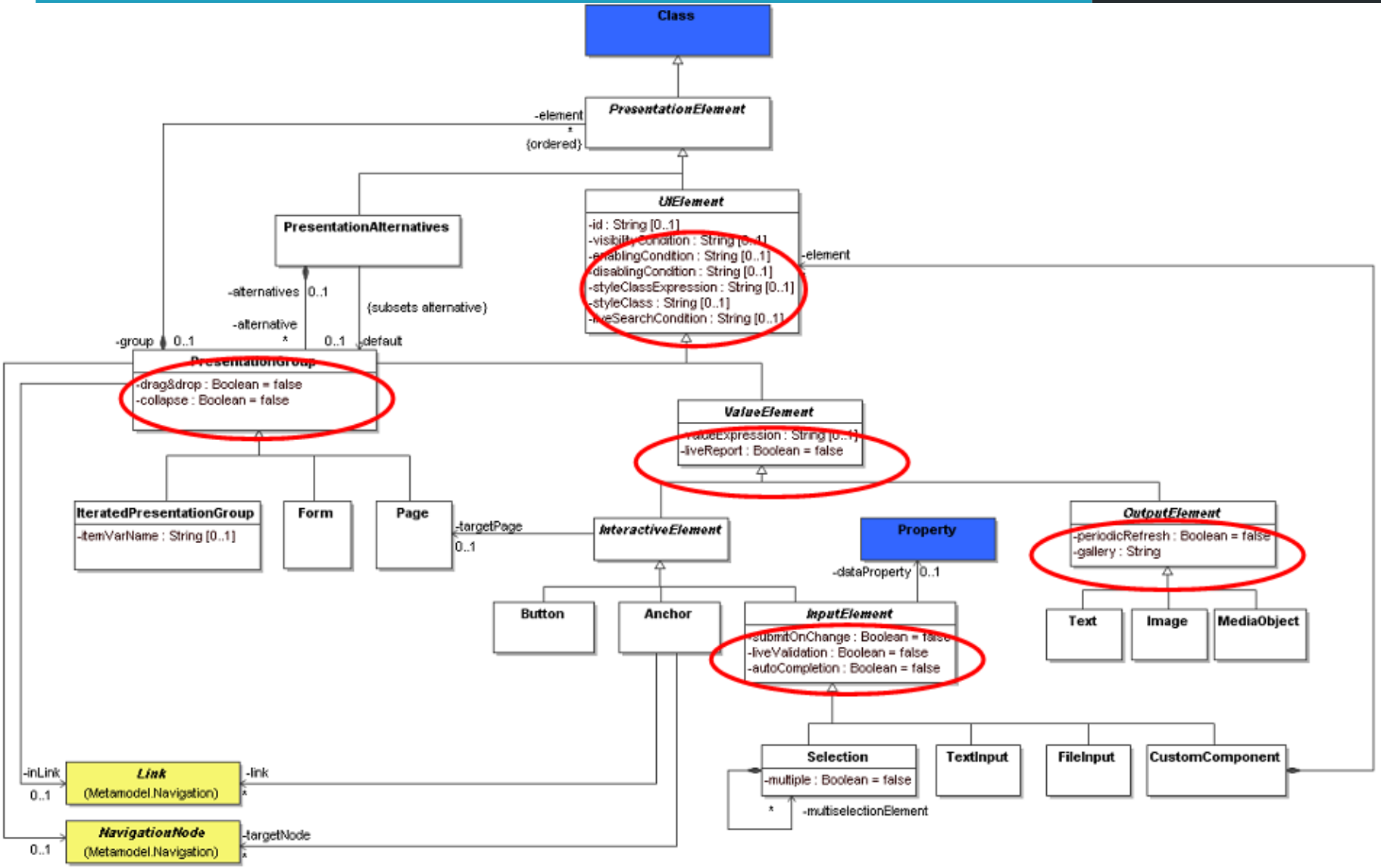




Navegación


Class
Association
NavigationNode
+isLandmark : Boolean
+isHome : Boolean
-guard : String [0..1]
-dataExpression : String [0..1]
-source
-target
1
1
*
*
-outLinks
-inLinks
Link
-isAutomatic : Boolean
-guard : String [0..1]
-selectionExpression : String [0..1]
ExternalNode
-locationExpression : String [1]
Menu
-menus
* 0..1
NavigationClass
NavigationLink
1
{subsets ownedAttribute}
-navigationProperty
*
NavigationProperty
-selectionExpression : String [0..1]
Property
AccessPrimitive
* -accessedAttributes *
Query
-expression : String [0..1]
GuidedTour
-sortExpression : String [0..1]
Index
IndexRowProperty

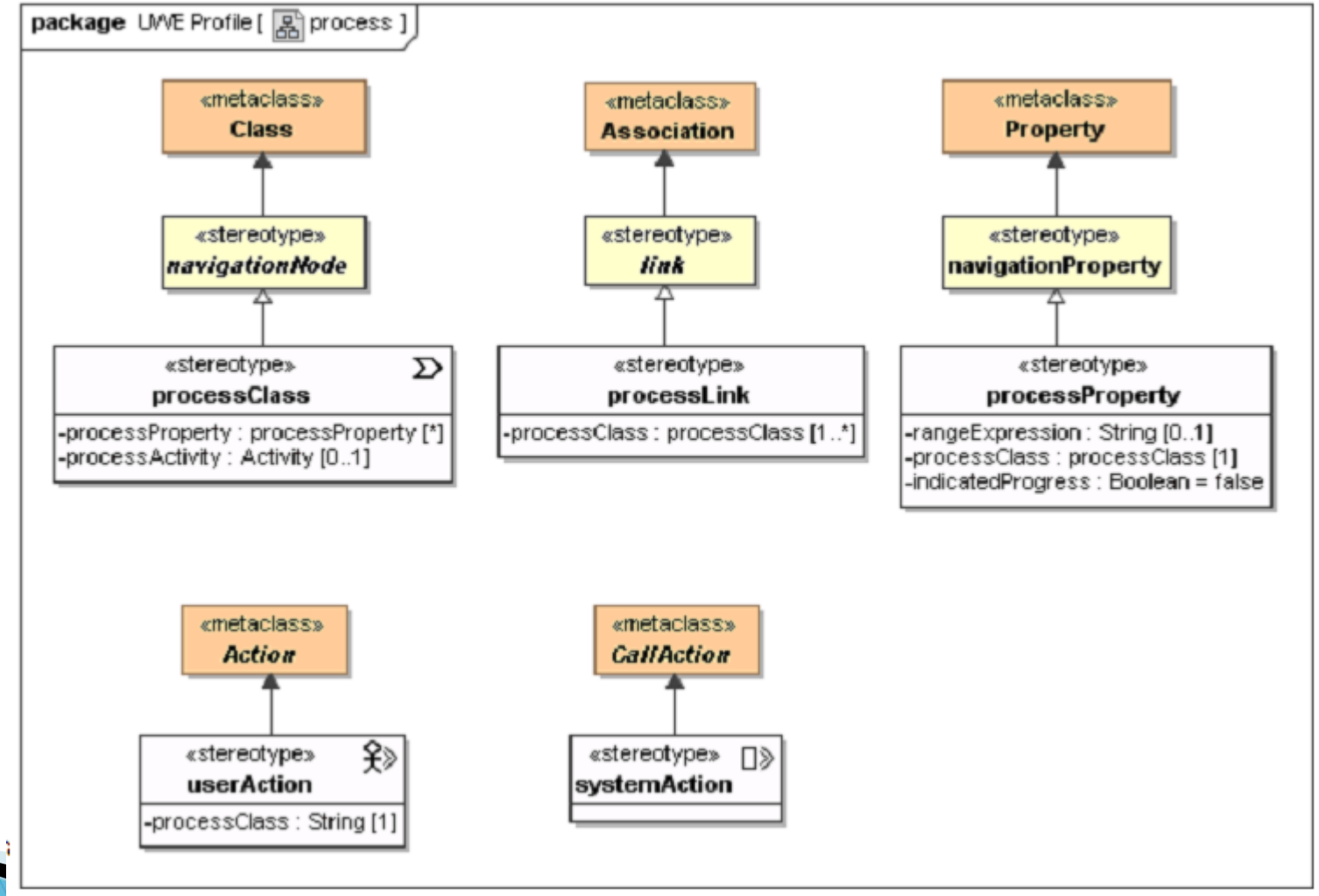
Presentación



Profile UWE

Profile

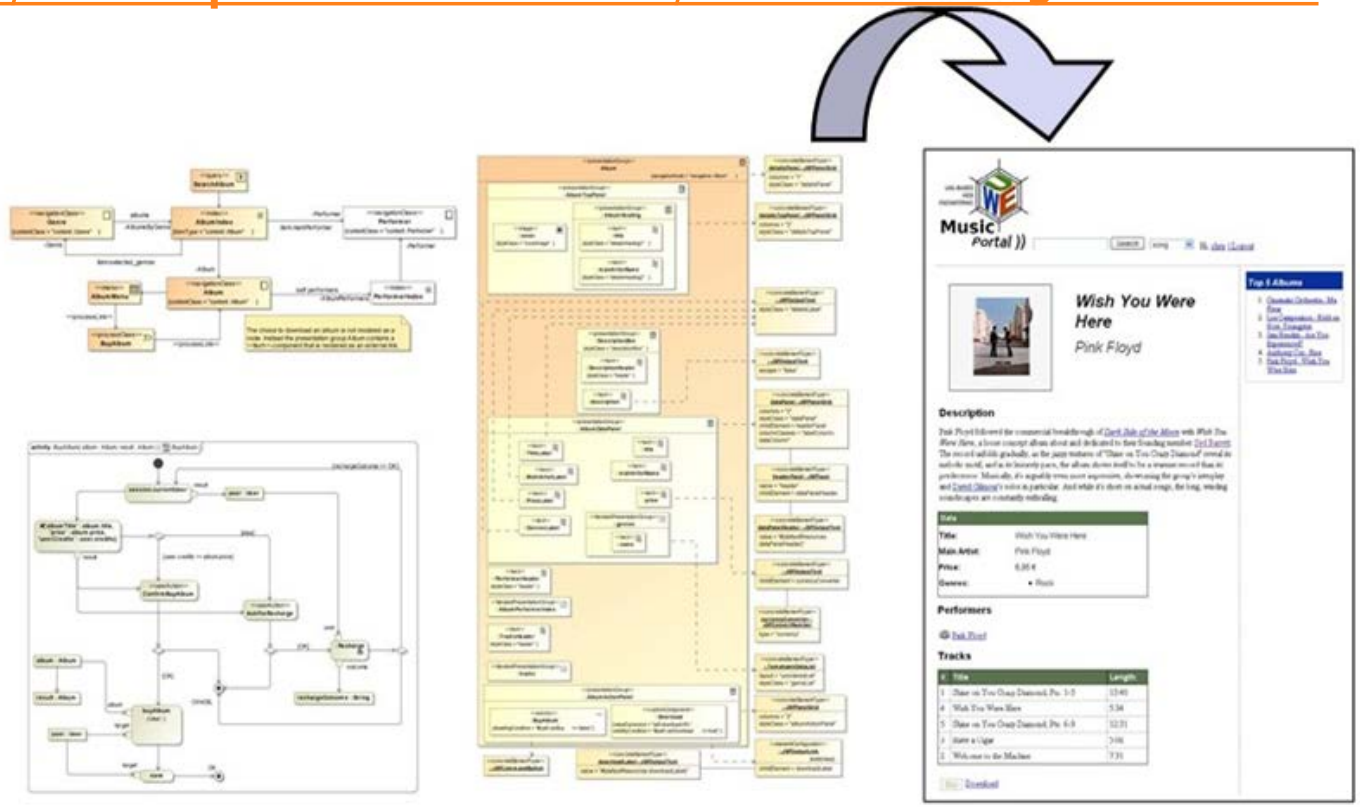
- Cada estereotipo extiende de una metaclase de UML



UWE4JSF:Herramienta para la generación de código

UWE4JSF

- Generación automática de aplicaciones RIA a partir de modelos UWE.
- http://uwe.pst.ifi.lmu.de/toolUWE4JSF.html



Módulo 4. Ingeniería Web. Modelado

Práctica Evaluable 1. UWE, MagicUWE

Se quiere tener una página web en la que se pueda administrar información sobre jugadores de fútbol, almacenados en un *Registro*. Las características de la página web que contendrá el registro de jugadores son las siguientes:

- La página principal del registro de jugadores debe tener un texto introductorio. El registro contendrá un conjunto de jugadores.
- Cada jugador debe tener información sobre su nombre y nacionalidad, ambos definidos como cadenas de caracteres. También tendrán información sobre su equipo actual (sólo uno), el último equipo en el que jugó (sólo uno), su coche principal (sólo uno), su coche especial (sólo uno) y su foto (sólo una).
- El equipo de los jugadores es especificado por su nombre, ciudad y país, todos ellos especificados por cadenas de caracteres. Como se ha detallado antes, cada jugador tendrá información relativa a su equipo actual y su último equipo.
- Los coches de los jugadores se especifican con su modelo, marca y color, los tres especificados con cadenas de caracteres. Cada jugador tiene un coche principal, al cual le da un uso diario, y un coche que reserva para ocasiones especiales.
- La foto de los jugadores debe tener información sobre su altura y anchura, especificada con enteros.

En cuanto a las acciones que el usuario podrá realizar sobre el registro de jugadores, son las siguientes:

- Buscar jugadores.
- Eliminar jugadores.
- Crear jugadores.
- Actualizar jugadores.
- Los cambios realizados sobre el registro podrán ser guardados o cancelados.

Se pide:

Modelar el registro web de jugadores utilizando *MagicUWE*. Para ello, se pide definir lo siguiente:

- Diagrama de Casos de Uso. Hay que crear además los estereotipos que se asignan a cada actividad, pues la herramienta *MagicUWE* no los ofrece.
- Diagrama de Contenido.
- Modelo de Navegación.
- Modelo de Presentación.

- Diagrama de Procesos, y al menos un Diagrama de Actividad para cada Proceso.
- Integración de Procesos en la Navegación.

Nota:

Esta práctica se debe hacer por grupos. Debe haber dos grupos de 4 personas y otro de 5. Se debe entregar un fichero PDF por cada grupo, donde aparecerán las oportunas capturas de pantalla de los distintos modelos, así como el texto aclaratorio que se desee añadir. Cada grupo se puede repartir el trabajo como quiera, pero la nota será la misma para todos los componentes del grupo. En el documento debe quedar explicitado el autor de cada uno de los modelos.