



# Tema 2. Análisis y Diseño Orientado a Objetos con UML


## Modelado del Software

Fernando Pereñíguez García

Escuela Politécnica



# Contenidos

- **Modelado del software** 
- Presentación de UML
- Objetivos de UML
- Conceptos de modelado
- Elementos de UML
- Diagramas en UML



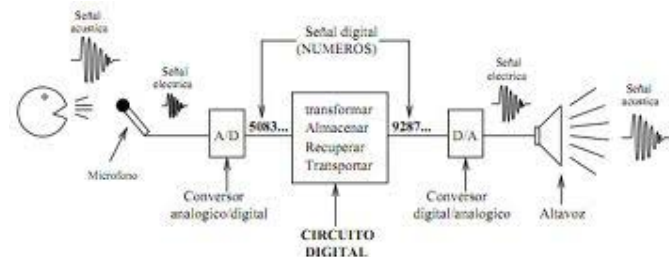
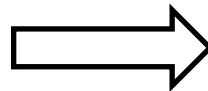
# La importancia de modelar

- ¿Qué es un modelo?
- ¿Para qué sirve un modelo?
- ¿Por qué es importante modelar?
- ¿Qué principios rigen el modelado del software?
- ¿Qué niveles de modelado existen?



# Concepto de modelo

- Un **modelo** es una **abstracción** de un sistema o entidad del mundo real
- Una **abstracción** es una **simplificación** de la realidad, que incluye sólo aquellos detalles relevantes para algún determinado propósito.
- El modelado permite **abordar la complejidad** de los sistemas.



# Objetivos del modelado

- Cuando se emplea el modelado de un sistema, se persiguen varios objetivos:
  - **Visualizar** cómo es o queremos que sea un sistema.
  - Permitir **especificar** la **estructura** o el **comportamiento** de un sistema.
  - Proporcionar **plantillas** que nos guían en la **construcción** de un sistema.
  - **Documentar** las **decisiones** que hemos adoptado



# ¿Por qué usar modelado?

- **Capturar y enumerar** exhaustivamente los **requisitos** y el dominio del conocimiento.
- **Pensar** en el **diseño** de un sistema.
- **Plasmar** las **decisiones de diseño** en un formato **alterable independiente** de los requisitos.
- Generar **productos usables para el trabajo**.
- Organizar, encontrar, filtrar, recuperar, examinar y corregir la **información** en **grandes sistemas**.
- **Explorar económicamente** múltiples soluciones.
- Dominar sistemas **complejos**.



# Principios del modelado

1. La **elección** acerca de qué **modelos** crear tiene una profunda influencia sobre cómo se acomete un problema y cómo se da forma a una solución.
2. Todo modelo puede ser expresado con diferentes **niveles de precisión**.
3. Los mejores modelos están ligados a la **realidad**.
4. Un único modelo o vista no es suficiente. Cualquier sistema no trivial se aborda mejor a través de un pequeño **conjunto de modelos** casi independientes con múltiples puntos de vista.

# Modelado del software (i)

- Uso del **código fuente**
  - Representa la lógica e ignora el resto.
  - Lento de procesar por el ser humano.
  - No facilita la reutilización ni la comunicación.
- Uso del **lenguaje natural**
  - Es lento de interpretar
  - Difícil de procesar de forma automática o semi-automática
  - Fácilmente puede dar lugar a confusión o ambigüedad.

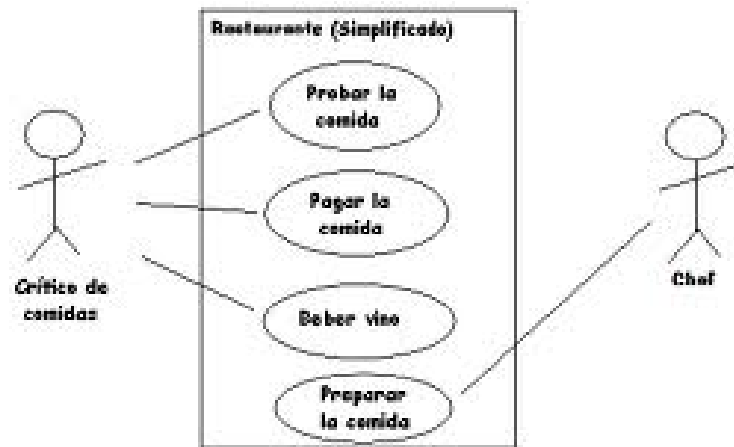
```
String sql = getStatement();  
resultSet = "select * from sto  
if (resultSet.next()) {  
    result = true;  
    setStoreId(resultSet.getInt("s  
    storeDescription = resu  
    storeTypeId = r  
    storeAdd
```

| Purchase orders   |
|---|
| Automatic creation of purchase orders from requisitions or RFQ'   |
| Fast manual creation of order from quotation request, requisition |
| Old order / part order duplication, plus edit facility            |
| System suggested item reorders, quantities, suppliers, with man   |
| System suggested best prices, suppliers by item, or default to la |
| Automatic tracking of item purchase price history                 |
| View recent purchases by item and by supplier                     |
| Multiple supplier volume discounts eg by order, spend amount, p   |
| breaks, per annum and dates available                             |
| Support and track multiple types of supplier rebates              |
| Automatic internet / extranet scan for latest / best prices       |



# Modelado del software (ii)

- Uso de **notación gráfica** (esquemas, diagramas, etc.)
  - Más fácil de evitar ambigüedad y confusión
  - Es fácil y rápido de interpretar
  - Es fácil de automatizar su procesamiento.



# Contenidos

- Modelado del software
- **Presentación de UML** 
- Objetivos de UML
- Conceptos de modelado
- Elementos de UML
- Diagramas en UML



# UML: Unified Modeling Language

- “*Lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema*”  
*(Booch, Jacobson y Rumbaugh)*
- Lenguaje de **modelado visual** de propósito general **orientado a objetos**.
  - Estandarizado por el *Object Management Group* (OMG)
- Es un **estándar independiente** de cualquier producto comercial
- Hoy en día, **ampliamente usado** en la industria del software



# Historia de UML

- Hasta la fecha había muchos modelos, metodologías y técnicas.
- UML nace como un esfuerzo para **simplificar** y consolidar el gran número **de métodos de desarrollo orientados a objetos**.
  - Agrupa notaciones y conceptos provenientes de diversos métodos Orientados a Objetos (OO)
- Mejor verlo de forma visual...



## Métodos para lenguajes tradicionales (70 - 80)

- Análisis y Diseño estructurado de Yourdon [Yourdon-79]
- También *Constantine*, *DeMarco*, *Mellor*,...

## Se popularizan los lenguajes O-O (80 - 90)

- Se popularizan los lenguajes OO
  - Simula-67 primer lenguaje OO
  - Smalltalk-80
  - Objective C, C++, Eiffel,...

## Métodos para lenguajes O-O (90 - 95)

- Desarrollo de multitud de métodos OO (Shlaer-Mellor, Coad-Yourdon,...)
- Conflicto entre diferentes propuestas
- Unos pocos empezaron a ganar importancia:
  - **Método de Booch**
  - **Método de Jacobson**: OOSE (Object-Oriented Software Engineering)
  - **Método de Rumbaugh**: OMT (Object Modelling Technique)

## Esfuerzo de unificación (95-96)

- Fusion (de Coleman) incluye conceptos de OMT de Rumbaugh, Método Booch y CRC (Wirfs-Brock)
- UML: primer éxito
  - 1994: Rumbaugh se une a Rational Software Corporation (RSC). Integración método Booch y OMT → UML 0.8
  - 1995: Jacobson se une a Rational Software Corporation. Extensión con método OOSE → UML 1.0

## Estandarización UML (97)

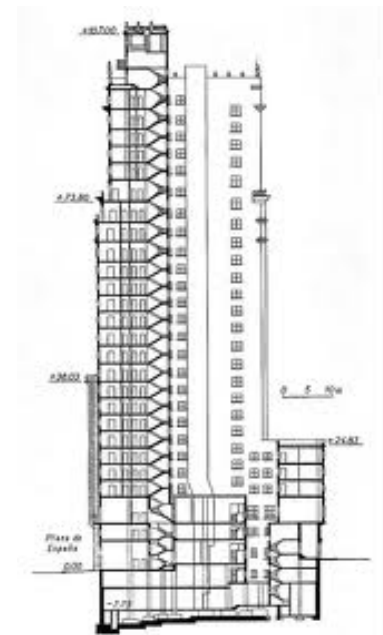
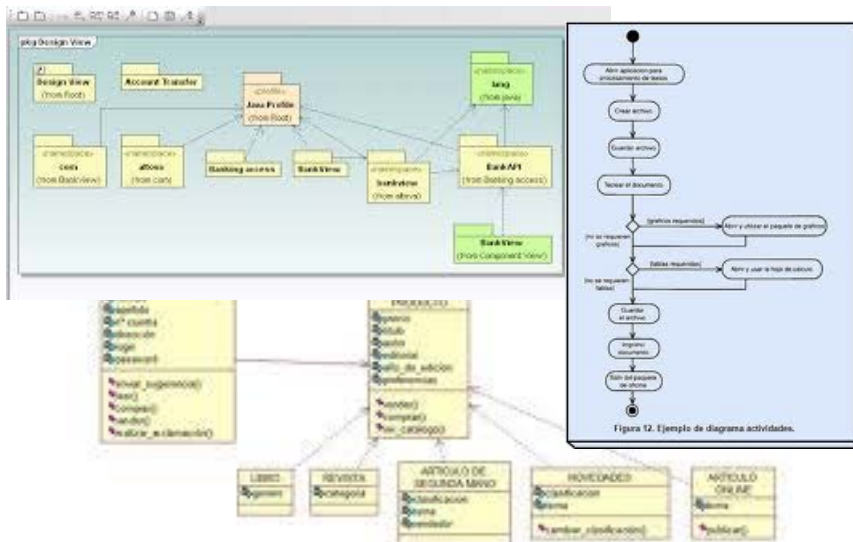
- UML se estandariza en el seno del OMG (*Object Management Group*)
- UML 1.1: primera versión estandarizada

## Estandarización UML (2000 – Actualidad)

- Evolución de UML: versiones 1.3, 1.4, 1.5
- En 2005 el OMG revisa UML y publica la versión 2.0

# Características de UML (i)

- **UML** proporciona un **vocabulario** y unas **reglas** que permitan:
  - Crear y entender modelos bien formados
  - Desarrollar los planos de un sistema sw.



## Características de UML (ii)

- **UML** es independiente del proceso de desarrollo
  - UML es sólo un lenguaje
  - Uso óptimo en procesos **iterativos** e **incrementales**.
    - Proceso Unificado de Desarrollo (RUP)
- **UML** permite expresar **todas las vistas** de un sistema software a lo largo de su ciclo de vida
  - Vistas estáticas
  - Vistas dinámicas





# Características de UML (iii)

- **Ventajas:**

- Es un **estándar**
- Semántica bien definida a través de un **metamodelo**
- **Notación gráfica** fácil de aprender y usar
- Aplicable para modelar sistemas software en **diversos dominios**
  - Sistemas empresariales
  - Sistemas de tiempo real
  - Sistemas web
  - Etc.
- Fácilmente **extensible**



# Características de UML (iii)

- No todo es perfecto...
- **Desventajas:**
  - UML **no es una metodología**
  - UML **no cubre todas las necesidades de especificación** de un proyecto (p.ej. no cubre el diseño de interfaces de usuario)
  - Puede resultar **complejo** alcanzar un conocimiento completo del lenguaje
    - Pensado para ser empleado en diferentes niveles y etapas del ciclo de vida del sistema software
    - Es la fusión de varias formas de modelar
  - Utiliza en exceso la **herencia**
  - Faltan **ejemplos** elaborados en la documentación que facilite su comprensión

# Contenidos

- Modelado del software
- Presentación de UML
- **Objetivos de UML** 
- Conceptos de modelado
- Elementos de UML
- Diagramas en UML

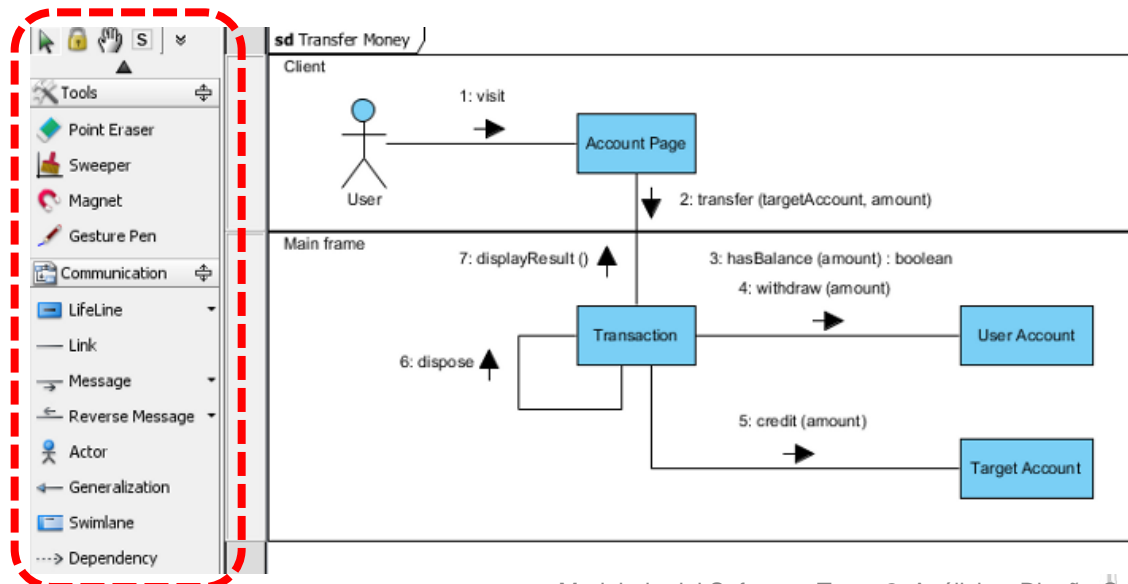


# Objetivos

- UML es un **lenguaje de modelado visual** que sirve para:
    - Visualizar
    - Especificar
    - Construir
    - Documentar
- ➡ sistemas software
- ➡ independientemente de la metodología de análisis y diseño .....
- ➡ pero siempre desde una perspectiva OO

# Objetivos – Visualizar

- UML proporciona:
  - **Notación**: un conjunto de símbolos
  - **Semántica**: da significado a cada símbolo
- Un modelo UML facilita la comunicación y puede ser representado en un lenguaje de programación.



# Objetivos – Especificar

- UML permite construir **modelos**:
  - Precisos
  - No ambiguos
  - Completos
- UML cubre las necesidades de especificación que puedan surgir en **diversas etapas** del desarrollo:
  - Recogida de requisitos
  - Análisis
  - Diseño
  - Implementación
  - Despliegue



# Objetivos – Construir

- UML permite la construcción de modelos software mediante un **lenguaje visual**
- UML **no** es una lenguaje de programación visual
- Sin embargo, es posible establecer **correspondencias** entre...
  - Modelo UML ↔ lenguaje de programación Java
  - Modelo UML ↔ bases de datos (relacionales OO)

# Objetivos – Documentar

- UML cubre toda la documentación de un sistema:
  - **Arquitectura** de un sistema y sus detalles
  - **Requisitos** y **pruebas**
  - Modelado de actividades de **planificación** y **gestión** de versiones

**El mantenimiento de un sistema depende de la calidad de la documentación**





# Contenidos

- Modelado del software
- Presentación de UML
- Objetivos de UML
- **Conceptos de modelado** 
- Elementos de UML
- Diagramas en UML



# Conceptos de Modelado

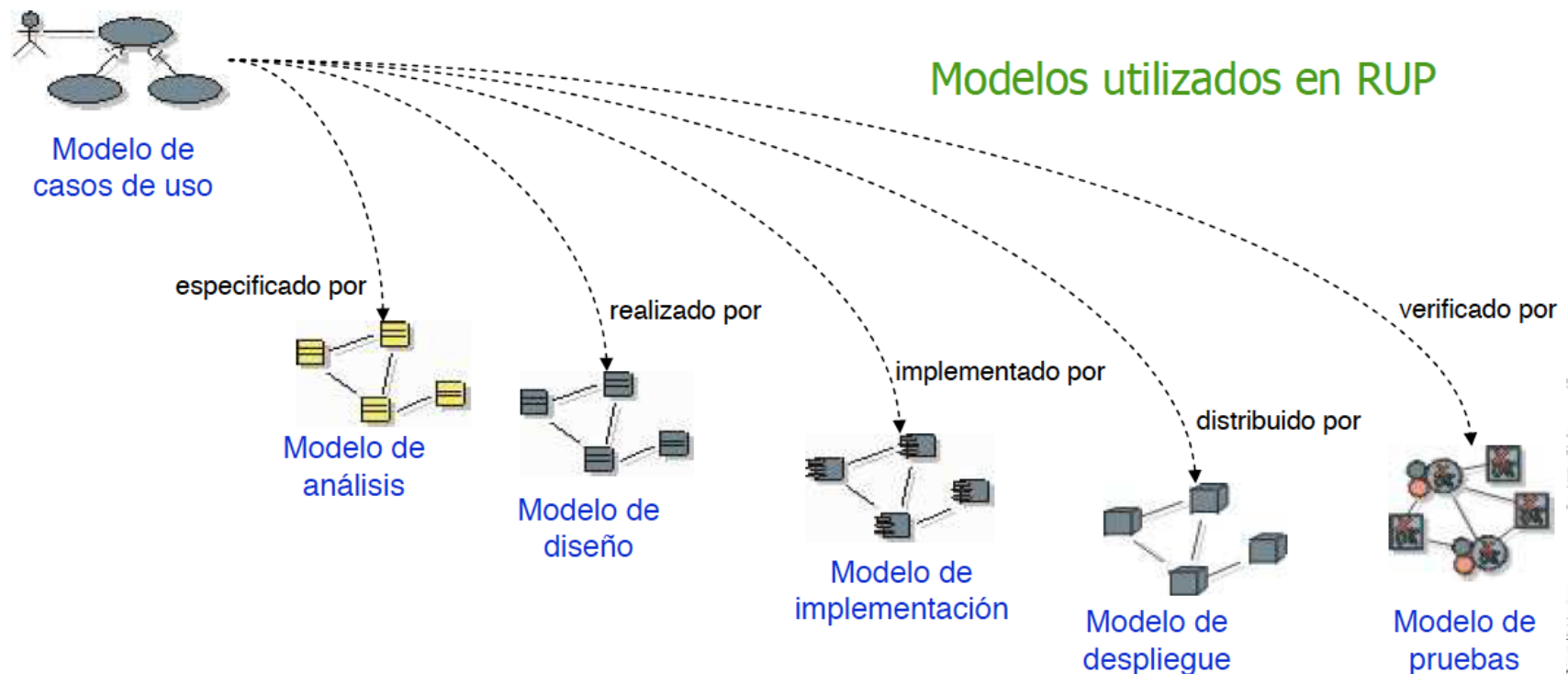
- **Sistema**
  - **Colección de elementos**, posiblemente divididos en subsistemas, organizados para **lograr un propósito**. Está descrito por un conjunto de modelos.
- **Modelo**
  - **Simplificación** completa y autoconsistente de la realidad, creado para **comprender mejor un sistema**.
- **Vista**
  - **Proyección** de la organización y estructura **de un modelo** de un sistema, centrada en un aspecto.
  - Incluye un subconjunto de los elementos de un modelo
- **Diagrama**
  - **Representación gráfica** de un conjunto de elementos del modelo y sus relaciones. En UML generalmente corresponde a un **grafo conexo de nodos** (elementos) y **arcos** (relaciones).

# Uso de modelos (i)

- Un modelo captura propiedades estructurales (**estáticas**) y de comportamiento (**dinámicas**) de un sistema.
- Un modelo describe **aspectos relevantes** de un sistema a un cierto **nivel de detalle**
  - El código fuente es el modelo más detallado que existe en un sistema.
- Los modelos no son independientes entre sí.
  - Existen relaciones de **trazabilidad** entre modelos.

# Uso de modelos (ii)

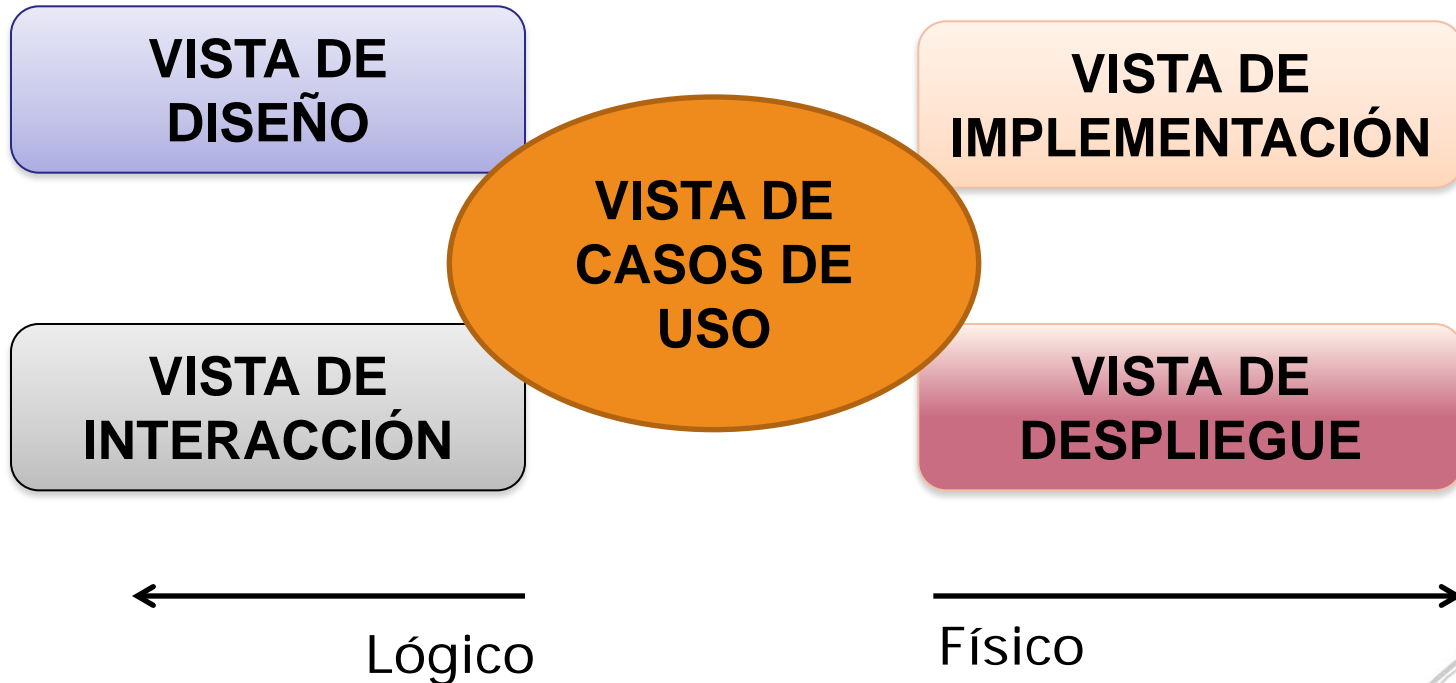
- Los modelos a utilizar los define la metodología que se aplique en el proceso de desarrollo.



# Vistas Arquitecturales

- Durante el desarrollo de un sistema software, se requiere que éste sea visto desde **varias perspectivas**.
  - Diferentes usuarios miran el sistema en distintos momentos (análisis, diseño, ...)
- **Vistas** de un sistema
  - Se establecen diferentes vistas arquitecturales. Cada vista se ocupa de analizar un **aspecto del sistema**
  - Una vista se compone de **un conjunto de modelos UML** que representan un aspecto del sistema software
- Las vistas están **interrelacionadas**

# Las 4+1 Vistas Arquitecturales



**Las vistas no forman parte de la especificación UML**

# Vista de casos de uso

- Captura la **funcionalidad** del sistema, tal y como es percibido por los usuarios finales, analistas y encargados de pruebas
- Describe la funcionalidad en base a **casos de uso**.
- En esta vista no se especifica la organización real del sistema software.
- Los **diagramas** que le corresponden son:

|                    | DIAGRAMAS   |
|--------------------|---|
| ASPECTOS ESTÁTICOS | Diagrama casos de uso   |
| ASPECTO DINÁMICOS  | Diagrama de interacción<br>Diagrama de estados<br>Diagrama de actividades |

# Vista de diseño

- Captura las **clases**, interfaces y colaboraciones que describen el sistema.
- Los elementos de esta vista dan soporte a los **requisitos funcionales**.
- Los **diagramas** que le corresponden son:

|                    | DIAGRAMAS   |
|--------------------|---|
| ASPECTOS ESTÁTICOS | Diagrama de clases y objetos  |
| ASPECTO DINÁMICOS  | Diagrama de interacción<br>Diagrama de estados<br>Diagrama de actividades |



# Vista de interacción

- Captura el **flujo de control** entre las diversas partes del sistema (p.ej. concurrencia, sincronización, etc.)
- Abarca en especial **requisitos no funcionales** como rendimiento, escalabilidad, etc.
- Los **diagramas** que le corresponden son los mismos que en la vista de diseño
  - Pero atendiendo a las clases activas que controlan el sistema y los mensajes entre ellas.



# Vista de implementación

- Captura los **artefactos** que se utilizan para ensamblar y poner en producción el sistema software.
- Define la arquitectura física, centrándose en aspectos como:
  - Configuración de los archivos físicos
  - Correspondencia entre clases y código fuente
  - Correspondencia entre componentes lógicos y artefactos físicos
- Los **diagramas** que le corresponden son:

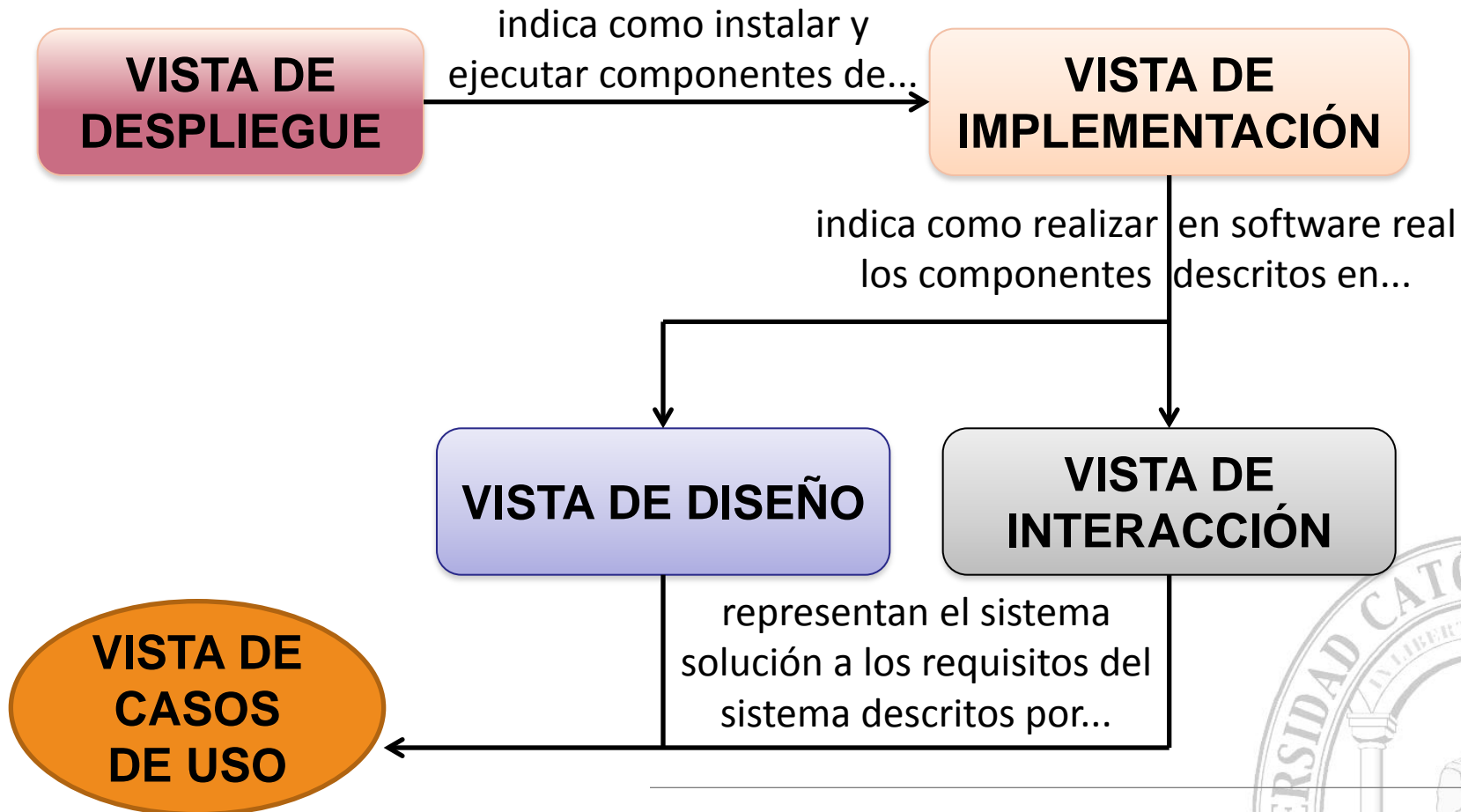
|                    | DIAGRAMAS   |
|--------------------|---|
| ASPECTOS ESTÁTICOS | Diagrama de componentes (artefactos)<br>Diagrama de estructura compuesta  |
| ASPECTO DINÁMICOS  | Diagrama de interacción<br>Diagrama de estados<br>Diagrama de actividades |

# Vista de despliegue

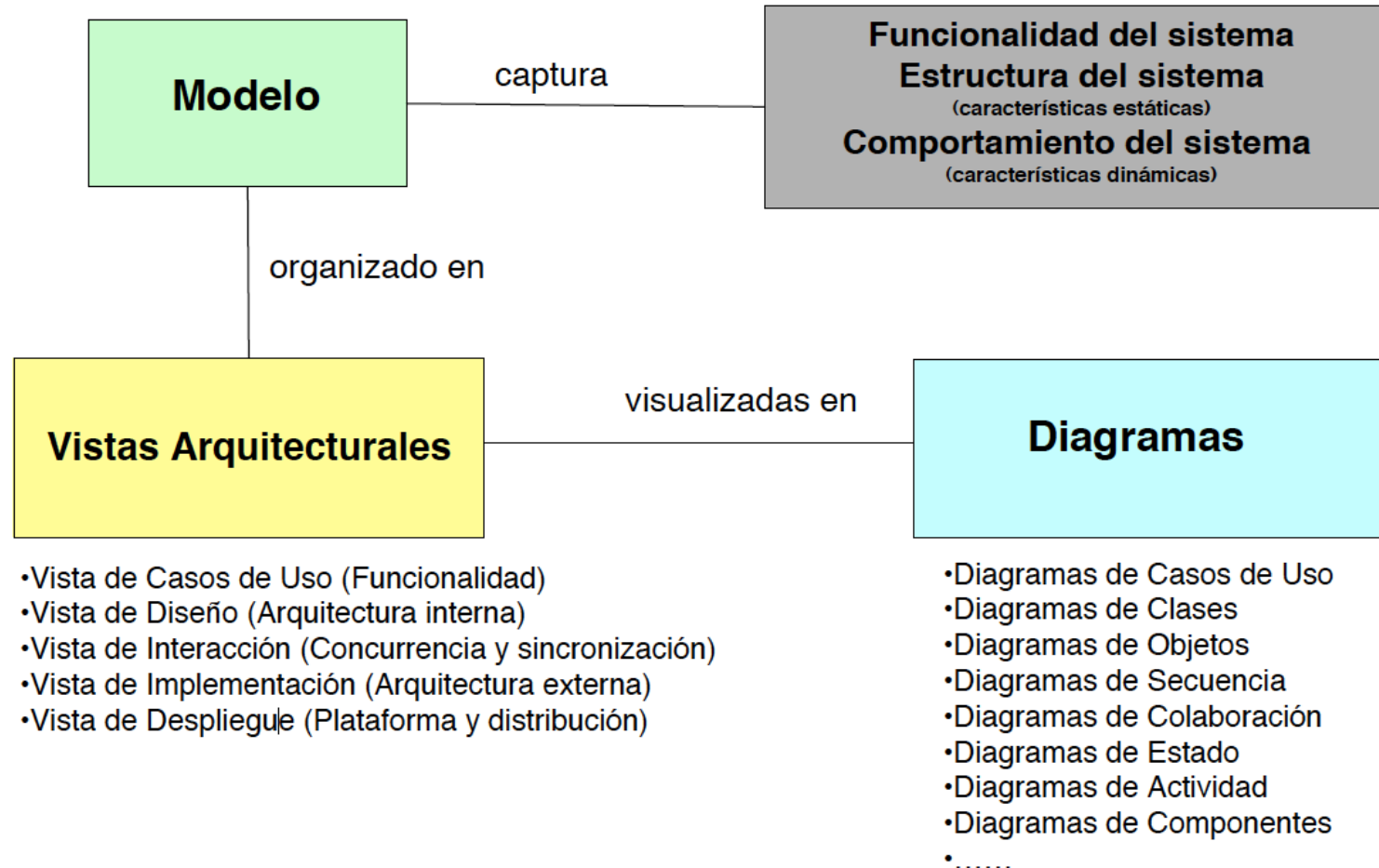
- Captura las características de **instalación y ejecución** del sistema software.
- Contiene los **nodos y enlaces** que forma la **topología hardware** sobre la que se ejecuta el sistema.
  - Especial énfasis en los componentes que forman el sistema software real.
- Los **diagramas** que le corresponden son:

|                    | DIAGRAMAS   |
|--------------------|---|
| ASPECTOS ESTÁTICOS | Diagrama de despliegue  |
| ASPECTO DINÁMICOS  | Diagrama de interacción<br>Diagrama de estados<br>Diagrama de actividades |

# Relación entre vistas

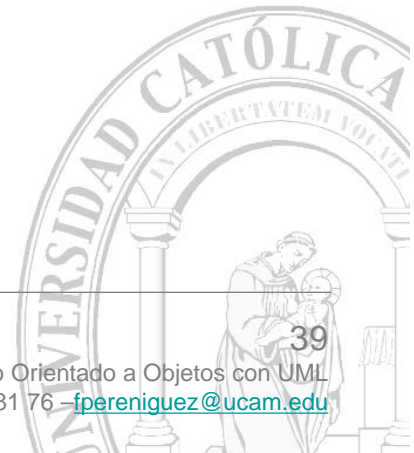


# Modelo UML de un sistema: Componentes



# Contenidos

- Modelado del software
- Presentación de UML
- Objetivos de UML
- Conceptos de modelado
- **Elementos de UML** 
- Diagramas en UML



# Elementos de UML

- La aplicación eficaz de UML requiere conocer y comprender su **metamodelo**.
- El metamodelo nos indica:
  - Los **elementos** que nos ofrece UML para modelar un sistema
  - El **significado** y uso de cada elemento
- El metamodelo de UML incluye tres tipos de elementos:
  - **Bloques de construcción**
  - **Reglas**
  - **Mecanismos comunes**



# Elementos de UML

## - Bloques de construcción

- Elementos
- Relaciones
- Diagramas

## - Reglas (indican cómo pueden combinarse los bloques)

## - Mecanismos comunes (aplicados al lenguaje)

- Especificaciones
- Adornos
- Divisiones comunes
- Mecanismos de extensibilidad



# Elementos de UML

## - Bloques de construcción

- Elementos
- Relaciones
- Diagramas

## - Reglas (indican cómo pueden combinarse los bloques)

## - Mecanismos comunes (aplicados al lenguaje)


- Especificaciones
- Adornos
- Divisiones comunes
- Mecanismos de extensibilidad

# Bloques de construcción - Elementos

- Son los bloques básicos para la construcción de un modelo UML
- Cuatro tipos de elementos:
  - Estructurales
  - De comportamiento
  - De agrupamiento
  - De anotación



# Bloques de construcción - Elementos

- Son los bloques básicos para la construcción de un modelo UML
- Cuatro tipos de elementos:
  - **Estructurales** 
  - De comportamiento
  - De agrupamiento
  - De anotación

# Bloques de construcción – Elementos estructurales

- Son los nombres de los modelos UML.
- En su mayoría, son las **partes estáticas** de un modelo.
- Representan conceptos o cosas materiales.
- En UML existen los siguientes tipos:

- Clase
- Interfaz
- Colaboración
- Caso de Uso
- Objeto
- Clase activa
- Componente

- Artefacto
- Nodo

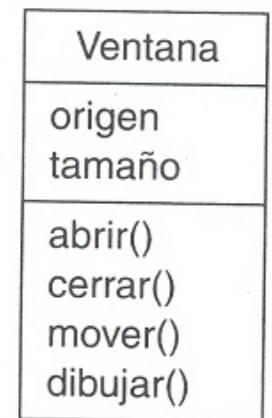
**Cosas lógicas**

**Cosas conceptuales**

# Bloques de construcción – Elementos estructurales

- **CLASE**

- Descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.
  - Una clase puede representar cosas hardware, personas, conceptos genéricos, etc.
- Es un concepto de diseño
  - En tiempo de ejecución, el sistema lo forman instancias de las clases (objetos)
- Las clases se usan en...
  - La fase de análisis: para capturar el vocabulario del sistema
  - La fase de diseño: para representar la solución software



# Bloques de construcción – Elementos estructurales

- **OBJETO**

- Representa una instancia de una clase en un determinado contexto
- Es un concepto relacionado con la ejecución del sistema software
  - El sistema lo conforman un conjunto de objetos que interaccionan entre sí.

bancoSantander:Banco

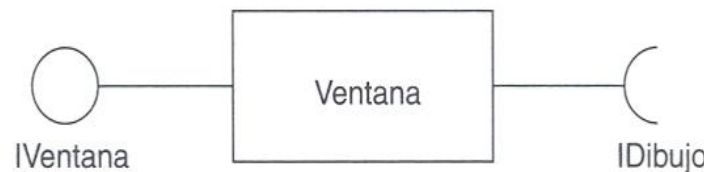
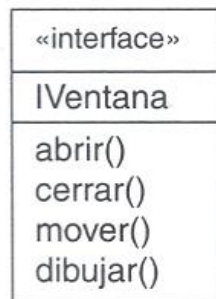
Pepe:Cliente

cuentaJuan:CuentaCliente

# Bloques de construcción – Elementos estructurales

## • INTERFAZ

- Colección de operaciones que especifican un servicio que puede ser ofrecido por una clase o componente
- Describen el comportamiento visible externamente de dichos elementos.
  - Puede representar el comportamiento completo o parcial del elemento.
- La interfaz hace visible las operaciones, nunca la implementación de las mismas.



# Bloques de construcción – Elementos estructurales

- CASO DE USO

- Describe el **comportamiento** de un sistema, clase o componente, desde el punto de vista del usuario.
- Describe un conjunto de secuencias de acciones que ejecuta un sistema y que produce un **resultado observable** que es de interés para un usuario particular.
- Se emplea para estructurar los aspectos de comportamiento.





# Bloques de construcción – Elementos estructurales

- **COLABORACIÓN**

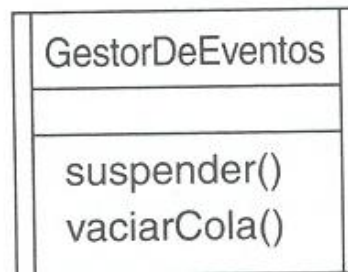
- Define una **interacción** entre elementos que colaboran para proporcionar un comportamiento mayor que de forma aislada.
- Una clase o un objeto puede participar en varias colaboraciones.



# Bloques de construcción – Elementos estructurales

- **CLASE ACTIVA**

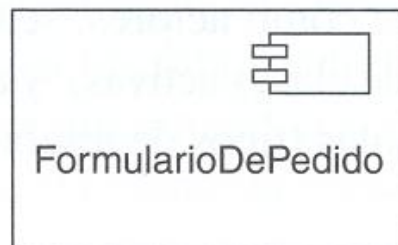
- Tipo especial de clase cuyos objetos tienen uno o más procesos o hilos de ejecución
  - Por tanto, puede dar lugar a actividades de control
- Son iguales que las clases, salvo que sus **objetos** pueden ser **concurrentes** con otros objetos de clases activas.

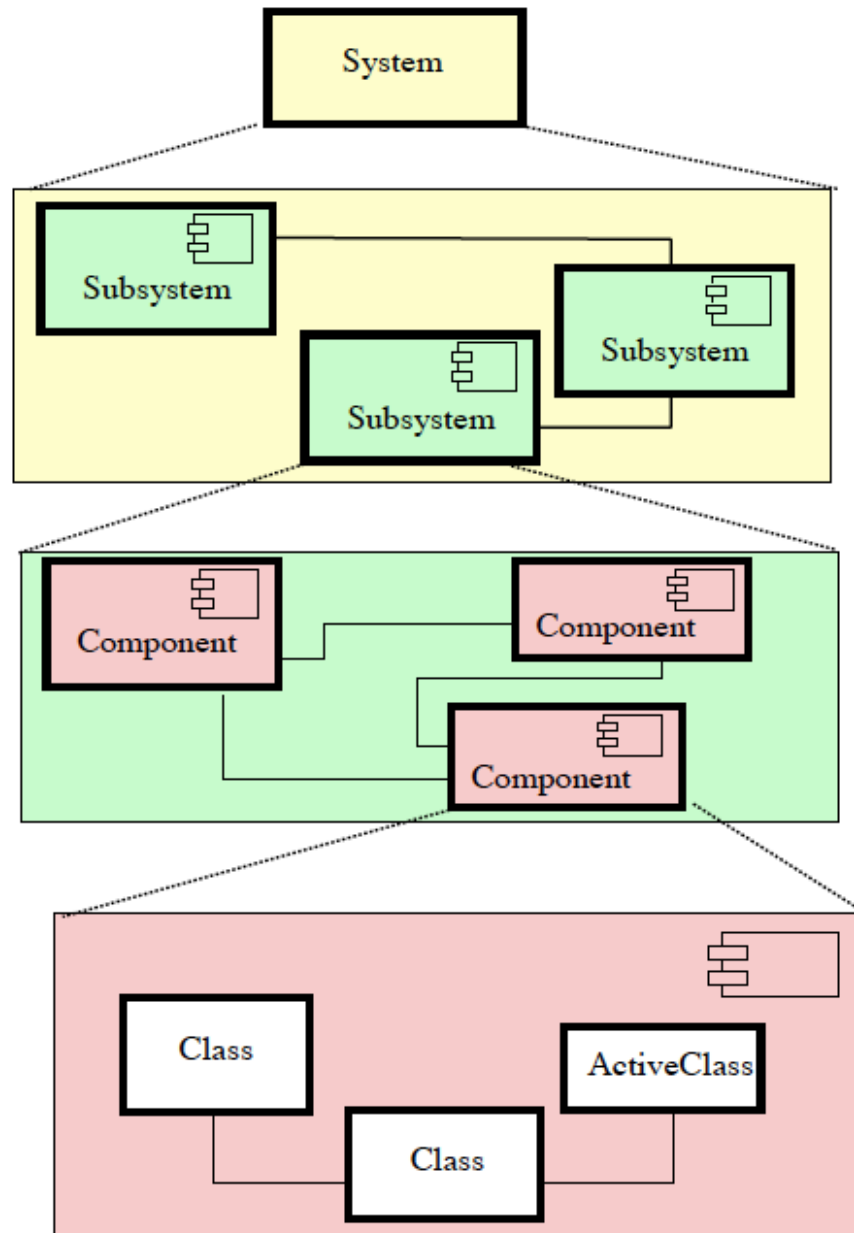


# Bloques de construcción – Elementos estructurales

- **COMPONENTE**

- Parte modular de la **arquitectura física** de un sistema
  - o Oculta implementación tras un conjunto de interfaces externas
- Su comportamiento se basa en **interfaces requeridas y ofertadas**
- **Empaquetamiento físico de diferentes elementos lógicos.**
- El sistema se define en base a componentes conectados entre sí.





# Bloques de construcción – Elementos estructurales

- **ARTEFACTO**

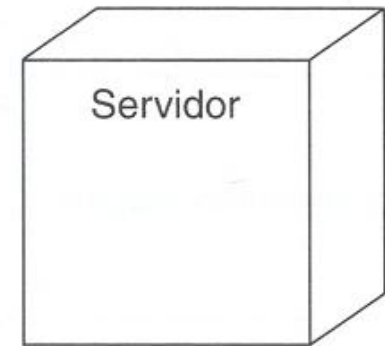
- Parte física de un sistema que contiene información (bits)
- Diferentes tipos de artefactos de despliegue:
  - Código fuente, ejecutables, scripts.




# Bloques de construcción – Elementos estructurales

- **NODO**

- Elemento físico que existe en tiempo de ejecución representa un **recurso computacional**
- Típicamente, un nodo **alberga un conjunto de artefactos**
- Los nodos describen las plataformas sobre las que se ejecuta el sistema software



# Bloques de construcción - Elementos

- Son los bloques básicos para la construcción de un modelo UML
- Cuatro tipos de elementos:
  - Estructurales
  - **De comportamiento** 
  - De agrupamiento
  - De anotación

# Bloques de construcción – Elementos de Comportamiento

- Son las **partes dinámicas** de los modelos UML
- Equivalen a los **verbos** de un modelo.
  - Representan el **comportamiento en el tiempo y en el espacio.**
- Suelen estar conectados semánticamente a los elementos estructurales
- Hay tres tipos:

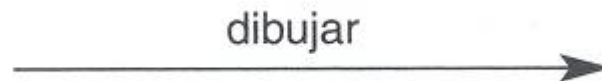
- Interacción
- Máquina de Estados
- Actividad



# Bloques de construcción – Elementos de comportamiento

## • INTERACCIÓN

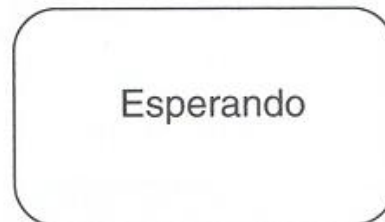
- Sirve para modelar el comportamiento de un conjunto de objetos
- Comportamiento que comprende un **conjunto de mensajes** intercambiados entre objetos:
  - o dentro de un contexto particular
  - o y para un propósito específico.
- Interacción entre objetos = **mensajes + acciones + enlaces**



# Bloques de construcción – Elementos de comportamiento

- **MÁQUINA DE ESTADOS**

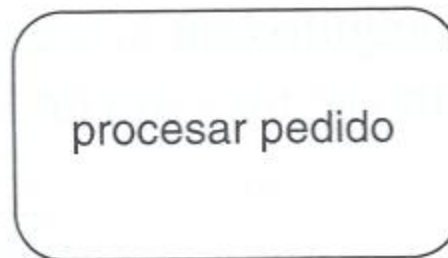
- Comportamiento que especifica **las secuencias de estados** por las que pasa un objeto o una interacción a lo largo de su vida, en respuesta a **eventos**, junto con sus **reacciones** a dichos eventos.
- Sirven para describir el comportamiento de una clase.
- Máquina estados = **estados + transiciones + actividades**



# Bloques de construcción – Elementos de comportamiento

- **ACTIVIDAD**


- Comportamiento que especifica la **secuencia de pasos** que ejecuta un proceso.
- Cada paso se le denomina **acción**.
- La notación para expresar una acción es la siguiente:



# Bloques de construcción – Elementos de comportamiento

- Resumiendo, tenemos tres alternativas para modelar el comportamiento...
  - En una **interacción**, el énfasis se pone en el conjunto de objetos que interactúan.
  - En una **máquina de estados**, el énfasis se pone en el ciclo de vida de un objeto.
  - En una **actividad**, el énfasis se pone en los flujos entre los pasos, sin mirar qué objeto ejecuta cada paso.

# Bloques de construcción - Elementos

- Son los bloques básicos para la construcción de un modelo UML
- Cuatro tipos de elementos:
  - Estructurales
  - De comportamiento
  - **De agrupamiento** 
  - De anotación

# Bloques de construcción – Elementos de Agrupamiento

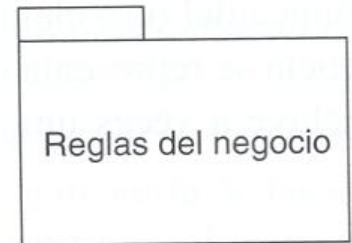
- Son las **partes organizativas** de los modelos UML
- Permiten expresar los componentes o “cajas” en que se puede dividir un modelo UML.
- Existe un tipo principal:

- Paquete

# Bloques de construcción – Elementos de Agrupamiento

- **PAQUETE**

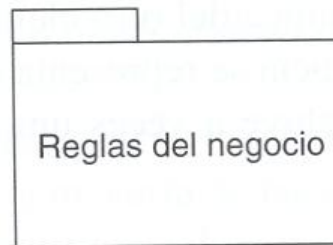
- Permite **organizar el diseño**.
  - Las clases nos permiten organizar construcciones de implementación.
- Puede incluir los elementos vistos anteriormente (estructurales y de comportamiento) así como otros paquetes.
- Concepto puramente conceptual: sólo existe en tiempo de desarrollo



# Bloques de construcción – Elementos de Agrupamiento


- **PAQUETE**

- Recomendaciones:
  - Debe contener elementos relacionados de forma lógica
  - Se puede utilizar en cualquier tipo de diagrama UML
  - Se puede controlar el nivel de visibilidad de los elementos de forma que algunos elementos sean visibles externamente y otros queden ocultos.



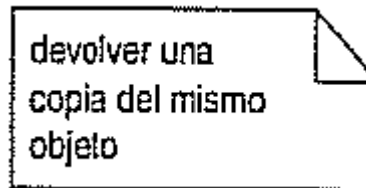


# Bloques de construcción - Elementos


- Son los bloques básicos para la construcción de un modelo UML
- Cuatro tipos de elementos:
  - Estructurales
  - De comportamiento
  - De agrupamiento
  - **De anotación** 

# Bloques de construcción – Elementos de Anotación

- Son las **partes explicativas** de los modelos UML
- Son comentarios que se añaden para describir, clarificar y hacer observaciones.
- Existe un tipo principal: **NOTA**
  - Símbolo para mostrar restricciones y comentarios asociados a un elemento o colección de elementos.
  - Empleado cuando es mejor hacer aclaraciones en texto formal o informal.



# Bloques de construcción - Relaciones

- Una **relación** es una conexión entre elementos estructurales.
- Cuatro tipos de relaciones en UML2:
  - Dependencia 
  - Asociación
    - Agregación
    - Composición
  - Generalización
  - Realización

# Bloques de construcción

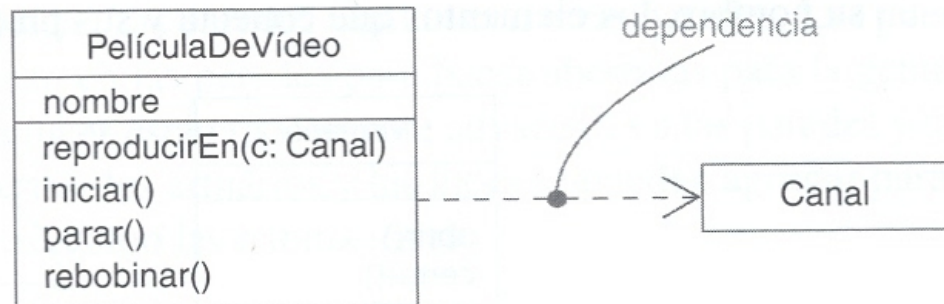
## Relaciones

- **DEPENDENCIA**


- Relación semántica en la cuál un cambio a un elemento puede afectar a la semántica de otro (dependiente)



- Uso frecuente entre clases
  - o Cuando una clase usa a otra como argumento en alguna operación



# Bloques de construcción - Relaciones

- Una **relación** es una conexión entre elementos estructurales.
- Cuatro tipos de relaciones en UML2:
  - Dependencia
  - Asociación 
    - Agregación
    - Composición
  - Generalización
  - Realización

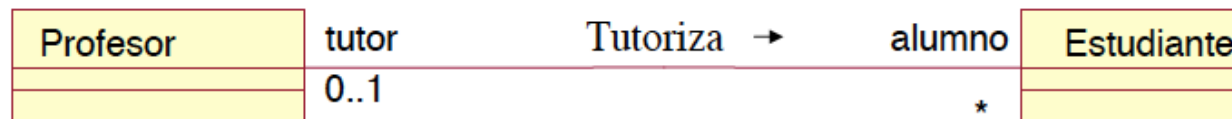


# Bloques de construcción

## Relaciones

- **ASOCIACIÓN**

- Relación estructural entre clases que describe un conjunto de enlaces (conexiones entre objetos que son instancias de clases)
  - Indicar que los **objetos** están **relacionados** durante un periodo de tiempo.
- Puede **identificarse por un nombre** que la describe (p.ej. un verbo)
- Otros adornos:
  - **Rol** que desempeña una clase en la asociación
  - **Multiplicidad** para la clase participante

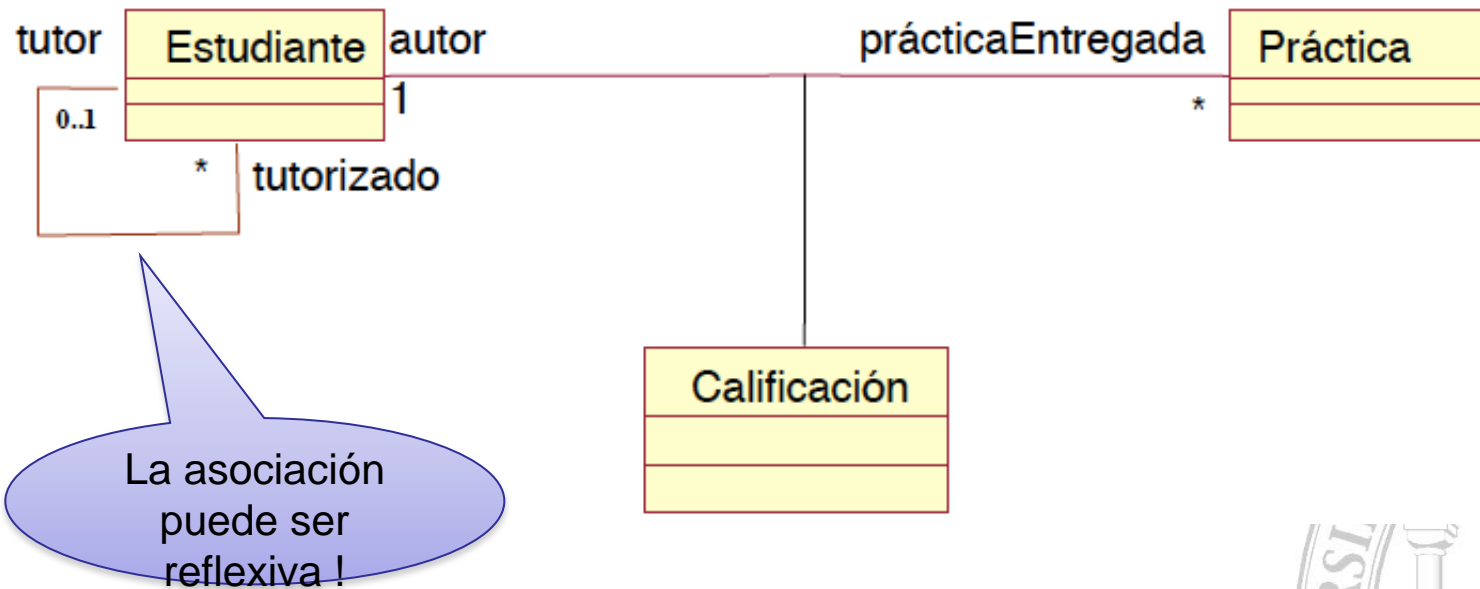


# Bloques de construcción

## Relaciones

- **ASOCIACIÓN**

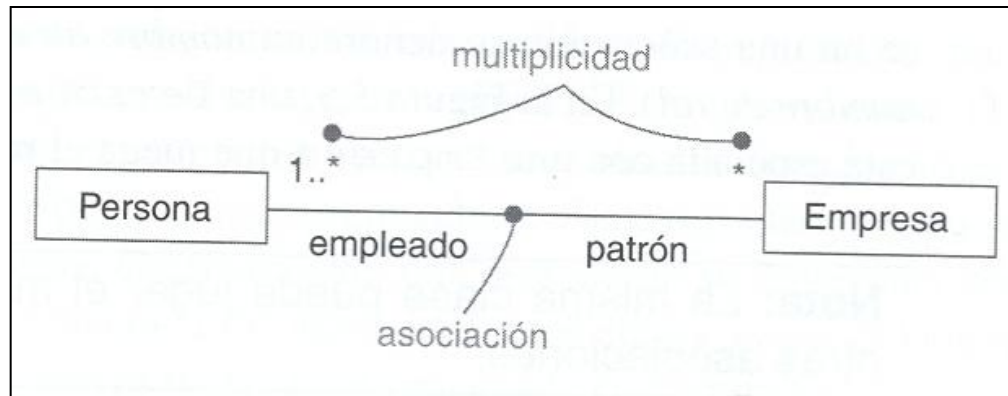
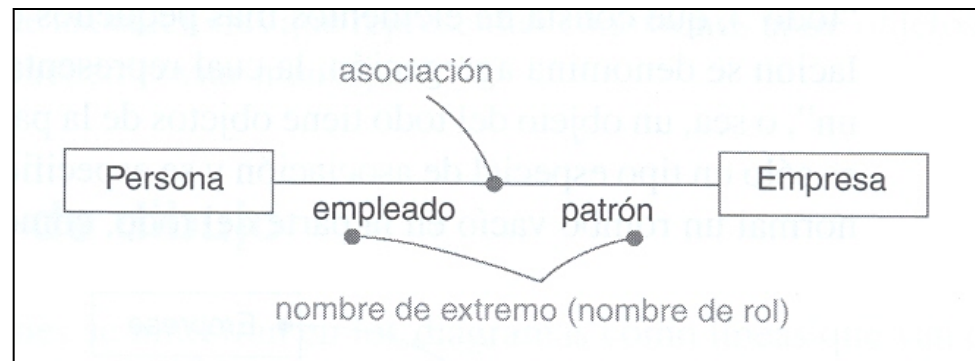
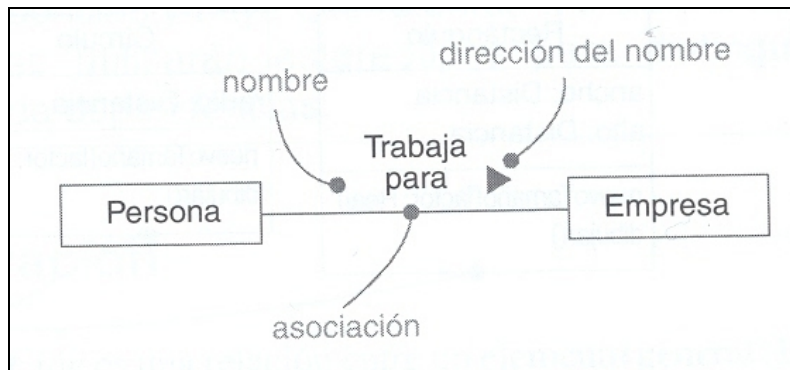
- Ejemplo de uso de asociación entre clases



# Bloques de construcción

## Relaciones

- **ASOCIACIÓN**
  - Más ejemplos:



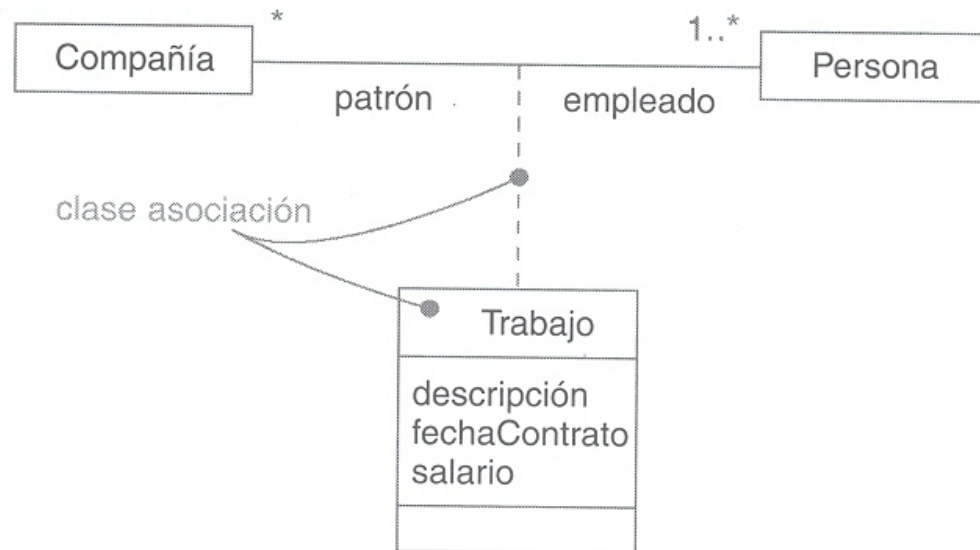


# Bloques de construcción

## Relaciones

- ASOCIACIÓN

- Ejemplos: uso de una **clase asociación**



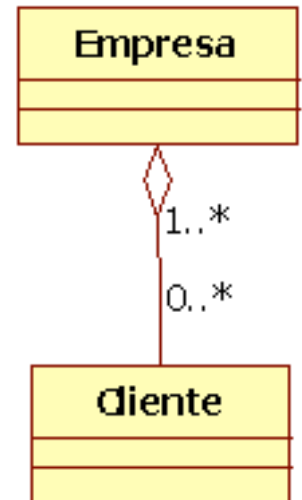
# Bloques de construcción

## Relaciones

- ASOCIACIÓN

- Agregación

- Indica que una clase es parte de otra clase
    - También llamada “composición débil”
    - Los componentes **pueden ser compartidos** por distintas asociaciones de agregación distintas.
    - La destrucción del compuesto no conlleva la destrucción de los componentes.
    - Símbolo: un diamante de color blanco en el extremo en el que está la clase que representa el “todo”.



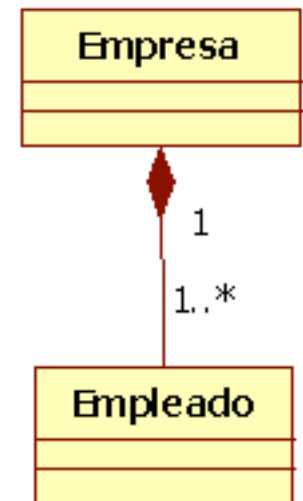
# Bloques de construcción

## Relaciones

- ASOCIACIÓN

- Composición

- También llamada “composición fuerte”
    - La vida de la clase contenida debe coincidir con la vida de la clase contenedor.
    - Los componentes constituyen una parte del objeto compuesto.
    - Los componentes **no pueden ser compartidos** por varios objetos compuestos.
      - La supresión del objeto compuesto conlleva la supresión de los componentes.
    - Símbolo: diamante de color negro en el extremo de la clase que representa el “todo”.



# Bloques de construcción

## Relaciones

- **ASOCIACIÓN**
  - Resumiendo...

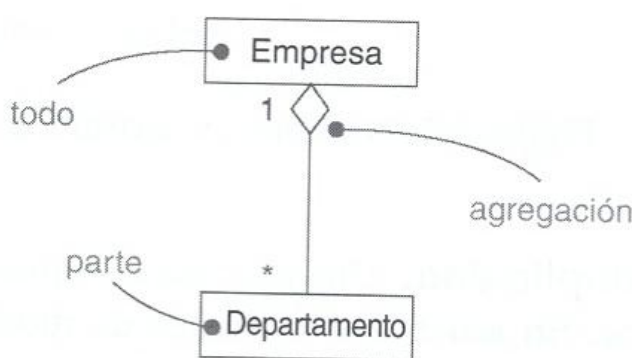
|   | <b>Agregación</b>  | <b>Composición</b> |
|---|--------------------|--------------------|
| Varias asociaciones comparten los componentes           | Sí                 | No                 |
| Destrucción de los componentes al destruir el compuesto | No                 | Sí                 |
| <u>Cardinalidad</u> a nivel de compuesto                | Cualquiera         | 0..1 ó 1           |
| Representación  | Rombo transparente | Rombo negro        |

# Bloques de construcción

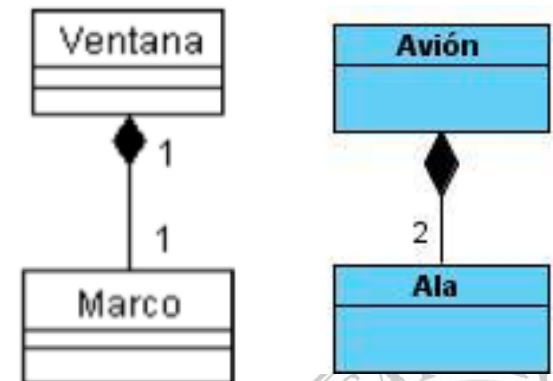
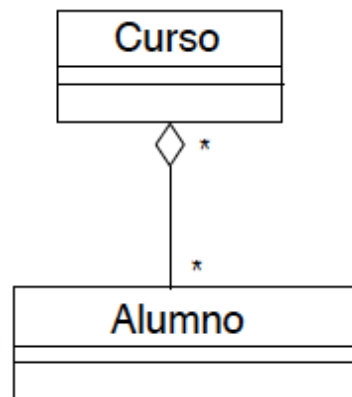
## Relaciones

### • ASOCIACIÓN

- Ejemplos de agregación y composición:



**AGREGACIÓN**



**COMPOSICIÓN**

# Bloques de construcción - Relaciones

- Una **relación** es una conexión entre elementos estructurales.
- Cuatro tipos de relaciones en UML2:
  - Dependencia
  - Asociación
    - Agregación
    - Composición
  - Generalización
  - Realización

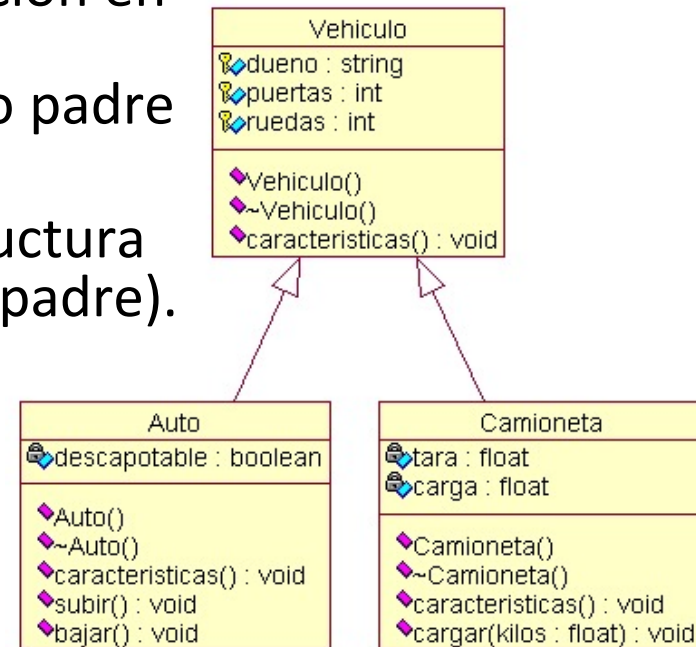
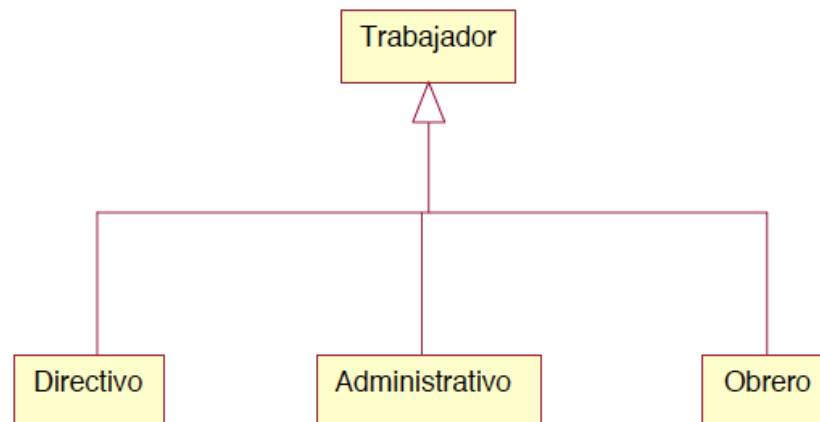


# Bloques de construcción

## Relaciones

- **GENERALIZACIÓN**

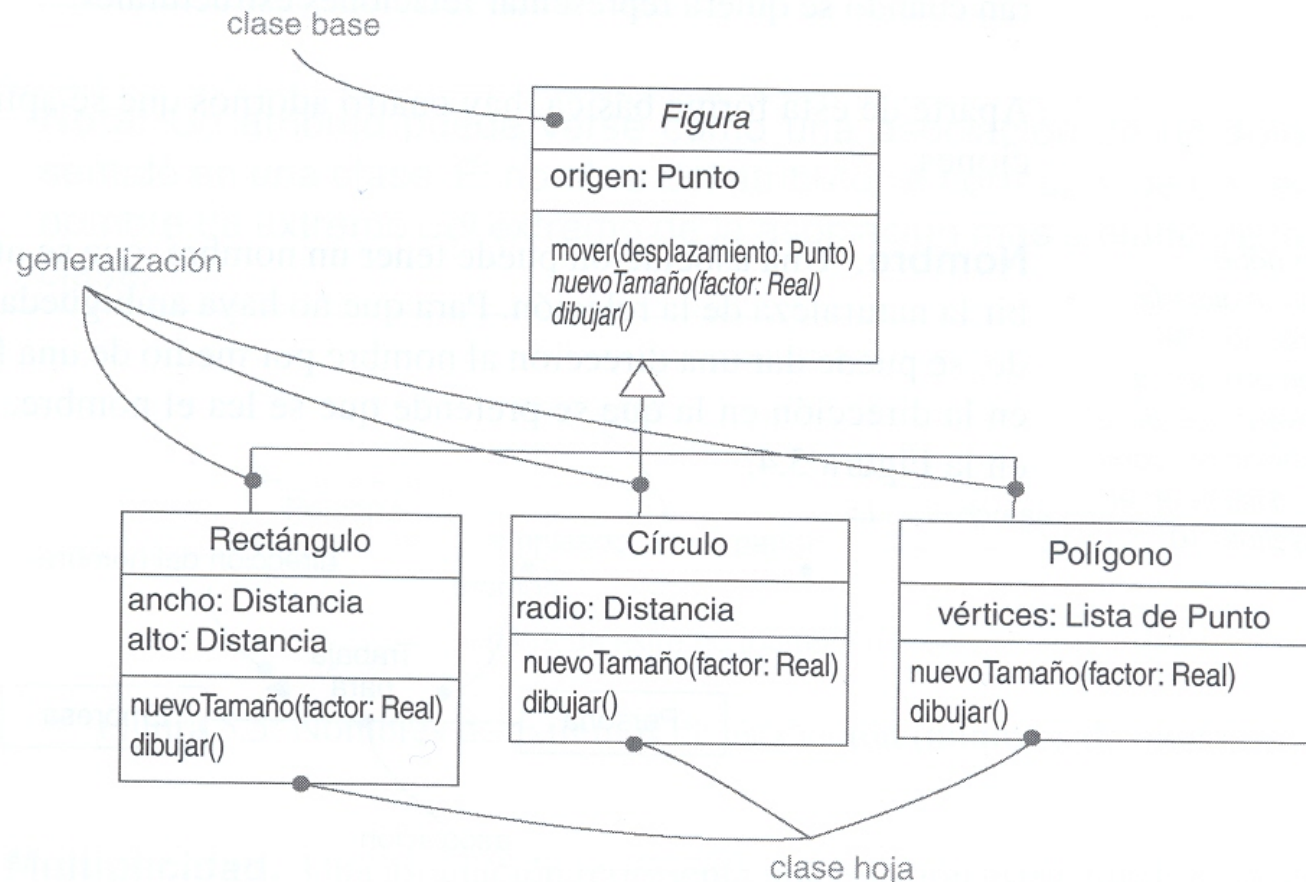
- Relación de especialización/generalización en la que el elemento hijo (especializado) extiende la especificación del elemento padre (generalizado)
- Las subclases (hijos) comparten la estructura y el comportamiento de la superclase (padre).



# Bloques de construcción

## Relaciones

- GENERALIZACIÓN





# Bloques de construcción - Relaciones

- Una **relación** es una conexión entre elementos estructurales.
- Cuatro tipos de relaciones en UML2:
  - Dependencia
  - Asociación
    - Agregación
    - Composición
  - Generalización
  - Realización

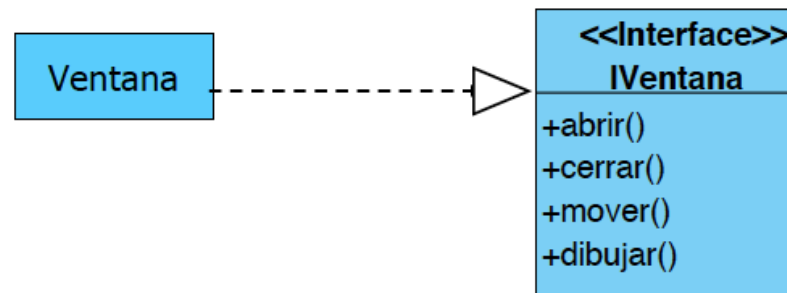


# Bloques de construcción

## Relaciones

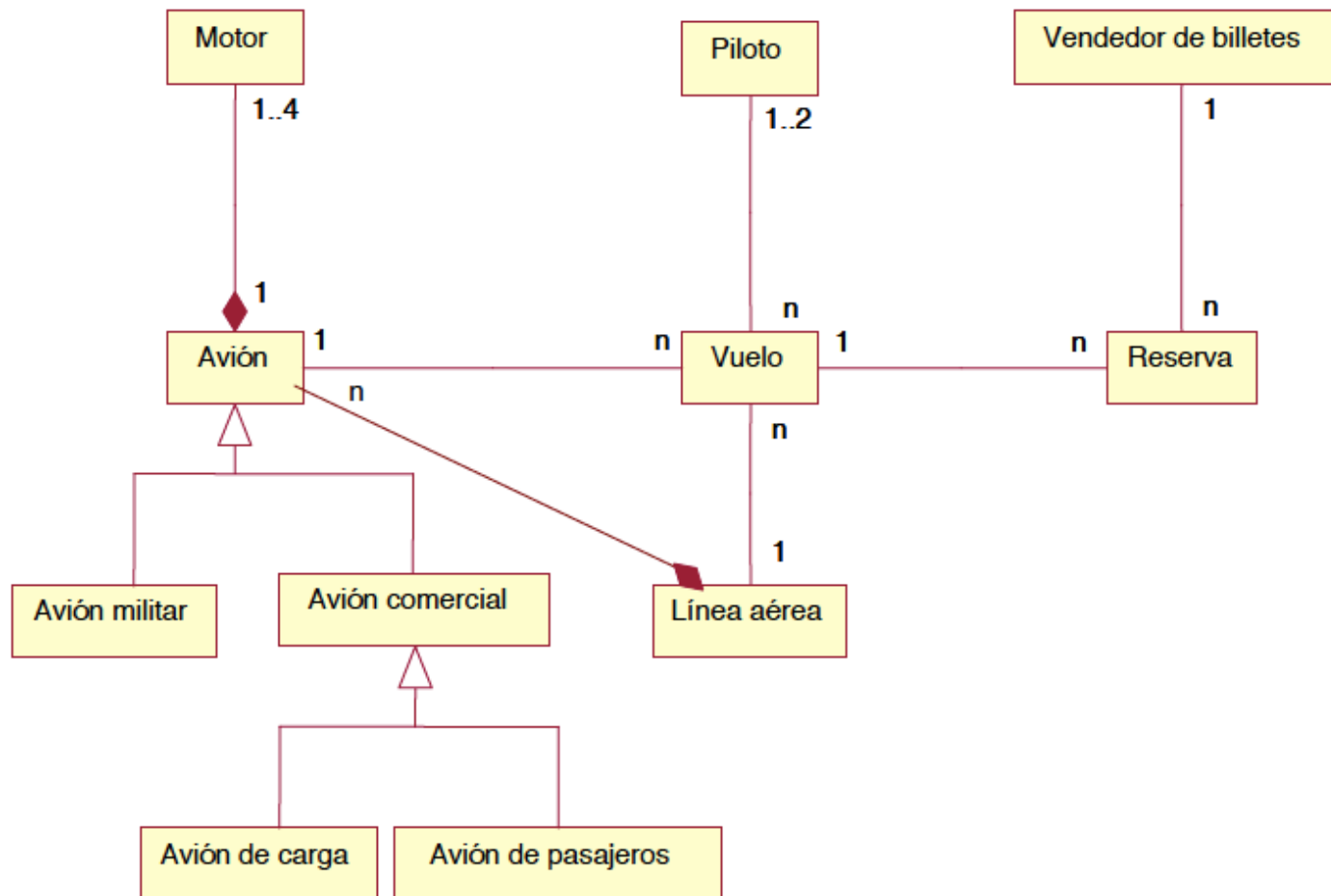
- **REALIZACIÓN**

- Relación semántica entre clasificadores, donde un clasificador especifica un **contrato** que otro clasificador garantiza que cumplirá.
- Empleados en dos situaciones:
  - Entre **interfaces y clases** que las realizan
  - Entre los **casos de uso y colaboraciones** que las realizan.



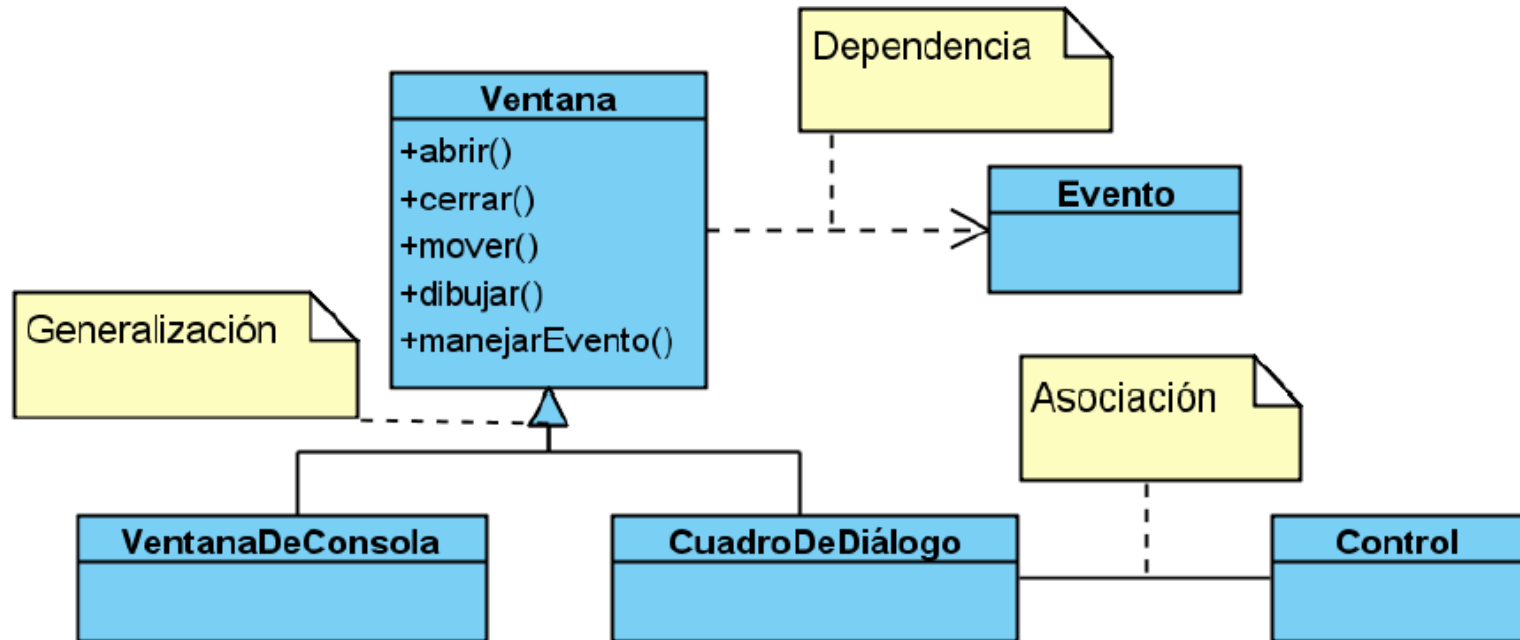
# Bloques de construcción

## Relaciones



# Bloques de construcción

## Relaciones



# Contenidos

- Modelado del software
- Presentación de UML
- Objetivos de UML
- Conceptos de modelado
- Elementos de UML
- **Diagramas en UML** 



# Bloques de construcción - Diagramas

- Un **diagrama** es una **presentación gráfica** de un conjunto de elementos, normalmente nodos y relaciones.
- Sirven para visualizar un sistema desde **diferentes perspectivas**.
- Un mismo elemento puede aparece en varios diagramas.
- En teoría, un diagrama puede contener cualquier combinación de elementos...
- Sin embargo, en la práctica, sólo un pequeño conjunto de combinaciones tienen sentido:

## TIPOS DE DIAGRAMAS DE UML2

# Bloques de construcción - Diagramas

- UML2 describe 13 tipos de diagramas

## ESTRUCTURALES (Estática)

- Clases
- Objetos
- Componentes
- Despliegue
- Paquetes
- Estructura compuesta

## DE COMPORTAMIENTO(Dinámica)

- Casos de uso
- Estados
- Actividades
- Interacción
  - Secuencia
  - Comunicación
  - Tiempos
  - Revisión de Interacciones

# Bloques de construcción

## Diagramas Estructurales

- Sirven para visualizar, especificar, construir y documentar **aspectos estáticos** del sistema.
- Se organizan en base a los grupos de elementos que aparecen al modelar.

| Diagrama             | Elemento                                 |
|----------------------|--|
| Clases               | Clases, interfaces y colaboraciones      |
| Componentes          | Componentes                              |
| Estructura Compuesta | Estructura interna de clase o componente |
| Objetos              | Objetos y sus relaciones                 |
| Paquetes             | Paquetes                                 |
| Despliegue           | Nodos                                    |



# Bloques de construcción

## Diagramas de Comportamiento

- Sirven para visualizar, especificar, construir y documentar **aspectos dinámicos** del sistema.
- Se organizan en base a las formas de modelar la dinámica de un sistema.

| Diagrama                  | Tipo de modelado dinámico   |
|---------------------------|---|
| Casos de Uso              | Especificar el comportamiento externo.                                  |
| Estados                   | Estado cambiante de los objetos dirigidos por modelos.                  |
| Actividades               | Flujo de control de actividades.  |
| Secuencia                 | Ordenación temporal de los mensajes.                                    |
| Comunicación              | Organización estructural de los objetos, envío y recepción de mensajes. |
| Tiempos                   | Tiempo real de los mensajes y estados.                                  |
| Revisión de interacciones | Vista general de las interacciones.                                     |

**Diagramas de Interacción**

# Elementos de UML

- Bloques de construcción

- Elementos
- Relaciones
- Diagramas

- **Reglas** (indican cómo pueden combinarse los bloques)

- Mecanismos comunes (aplicados al lenguaje)

- Especificaciones
- Adornos
- Divisiones comunes
- Mecanismos de extensibilidad

# Reglas

- Los distintos bloques de construcción no pueden combinarse de cualquier manera.
- UML tiene reglas para obtener un **modelo bien formado**:
  - Semánticamente autoconsistente
  - En armonía con los modelos relacionados
- También se admiten modelos de sistemas:
  - **Abreviados**: Ciertos elementos se ocultan para simplificar la vista.
  - **Incompletos**: Pueden estar ausentes ciertos elementos.
  - **Inconsistentes**: No se garantiza la integridad del modelo

# Reglas sintácticas y semánticas

- UML define reglas para:
  - **Nombres:** cómo llamar a los elementos, relaciones y diagramas.
  - **Alcance:** determinar el contexto que da un significado específico a un nombre.
  - **Visibilidad:** determinar cómo se pueden ver y utilizar esos nombres por otros.
  - **Integridad:** describir cómo se relacionan apropiada y consistentemente unos elementos con otros.
  - **Ejecución:** indicar el significado de ejecutar o simular un modelo dinámico.

# Elementos de UML

- **Bloques de construcción**

- Elementos
- Relaciones
- Diagramas

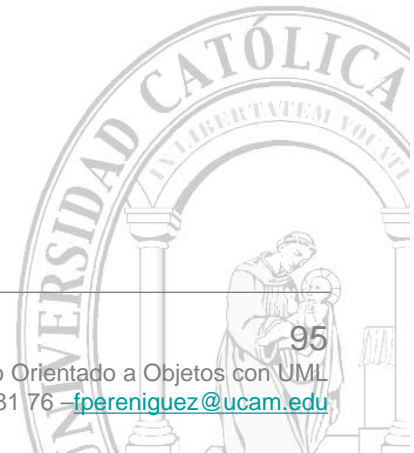
- **Reglas** (indican cómo pueden combinarse los bloques)

- **Mecanismos comunes** (aplicados al lenguaje)

- Especificaciones
- Adornos
- Divisiones comunes
- Mecanismos de extensibilidad

# Mecanismos comunes

- UML tiene cuatro mecanismos comunes que se aplican de forma consistente a través de todo el lenguaje
  - Especificaciones
  - Adornos
  - Divisiones comunes
  - Extensibilidad
- Se aplican a **todos** los bloques de construcción
- Beneficios:
  - Dan **cohesión**
  - **Simplifican** el modelo



# Mecanismos comunes - Especificaciones

- UML no es sólo un lenguaje gráfico
- Detrás de cada elemento gráfico hay una especificación que explica la sintaxis y semántica.
  - La **especificación** se utiliza para expresar los detalles de dicho sistema.
  - Los **notación gráfica** son proyecciones visuales de la base semántica provista por las especificaciones UML.
- Todo elemento gráfico de un diagrama UML tiene una especificación asociada: atributos, operaciones, etc.
  - Visualmente el icono **sólo muestra parte de la información**.

# Mecanismos comunes - Especificaciones

The diagram illustrates the mapping between a UML class and its corresponding 'Class Specification' dialog box. The UML class 'Vehiculo' is shown on the left with attributes -marca and -modelo, and operation +acelerar(). The 'Class Specification' dialog box is shown in the center, with tabs for Constraints, Diagrams, References, and Comments. The 'General' tab is selected, showing the Name: Vehiculo, Parent: <None>, Visibility: public, and Documentation: HTML. The 'Attributes' tab is also visible, showing the attributes -marca and -modelo. The 'Operations' tab is also visible, showing the operation +acelerar(). The 'References' tab is also visible, showing the operation +acelerar(). The 'Comments' tab is also visible, showing the operation +acelerar().

**Vehiculo**

- marca
- modelo
- +acelerar()

**Class Specification**

Constraints | Diagrams | References | Comments

Class Code Details | ORM Query | Stereotypes | Tagged Values

General | Attributes | Operations | Relations | Template Parameters

Name: Vehiculo

Parent: <None>

Visibility: public

Documentation: HTML

HTML | B | I | U | L | F | Fr | ...

☐ Abstract ☐ Leaf ☐ Root ☐ Active ☐ Business model

Reset OK Cancel Apply Help

**Vehiculo**

- marca: String[1]
- modelo: String[1]
- + acelerar(int kmh)



# Mecanismos comunes - Adornos

- Todos los **elementos** en la notación gráfica de UML parte de un **símbolo básico** que proporciona una representación visual de los **aspectos más importantes**.
- Para incluir más detalles, se pueden usar **adornos**.
- Ejemplo de adornos de una clase:
  - Si es abstracta o no, visibilidad de atributos y operaciones, tipos de atributos, etc.



(+) Público  
(#) Protegido  
(-) Privado

# Mecanismos comunes - Divisiones

- En el modelado orientado a objetos, existen varias divisiones comunes del mundo.
- **Clase - Objeto**
  - Una clase es una **abstracción**
  - Un objeto es una **manifestación concreta** de dicha abstracción.

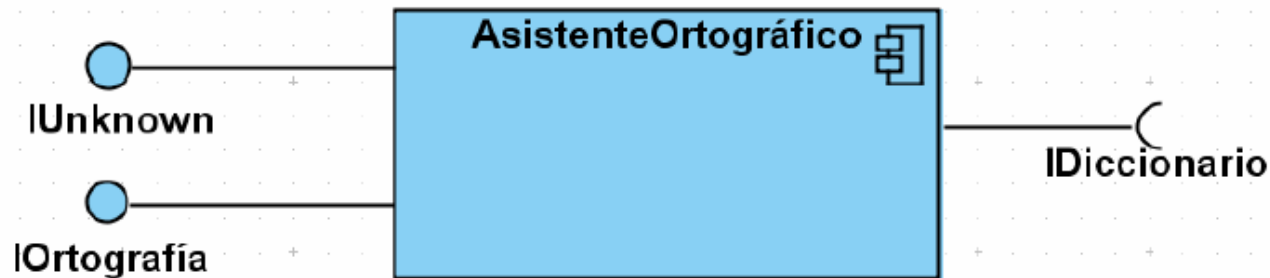


- Otros ejemplos: **componente – instancia de componente**, **nodo – instancia de nodo**, etc.

# Mecanismos comunes - Divisiones

- **Interfaz - Implementación**

- Una interfaz declara un **contrato**
- Una implementación representa una **realización concreta** de ese contrato

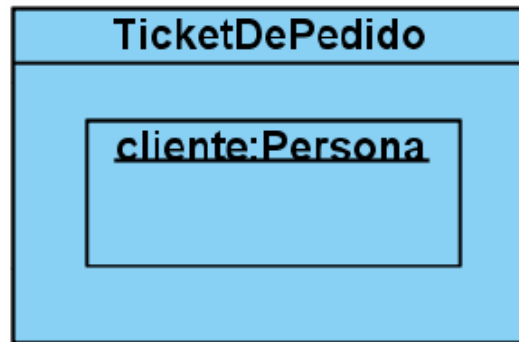


- Otros ejemplos: **caso de uso – colaboración que la realizan, operación – métodos que las implementan, etc.**

# Mecanismos comunes - Divisiones

- **Tipo - Rol**

- Un tipo declara la **clase de una entidad** (objeto, atributo,...)
- Un rol describe el **significado** de una entidad **en un contexto** (p.ej. en una clase, componente o colaboración)



Atributo de tipo  
"Persona"

Juega el rol "cliente" en el  
Ticket de un pedido

# Mecanismos comunes - Extensibilidad

- UML se puede extender en base a la definición de **perfiles**.
  - Un perfil es una extensión de UML para poder expresar todos los **matices** de un sistema software en un **determinado dominio**.
- Un perfil comprende tres componentes:
  - **Estereotipo**: para añadir nuevos bloques de construcción
  - **Valores etiquetados**: para modificar la especificación de los nuevos bloques de construcción
  - **Restricciones**: cambiar o añadir una semántica a un elemento de modelado.

# Mecanismos comunes - Extensibilidad

- **Estereotipo**

- Extiende el vocabulario de UML permitiendo crear **nuevos** tipos de **bloques de construcción** que derivan de los existentes pero que son específicos a un problema.
- Ejemplo: las excepciones de Java son clases especiales

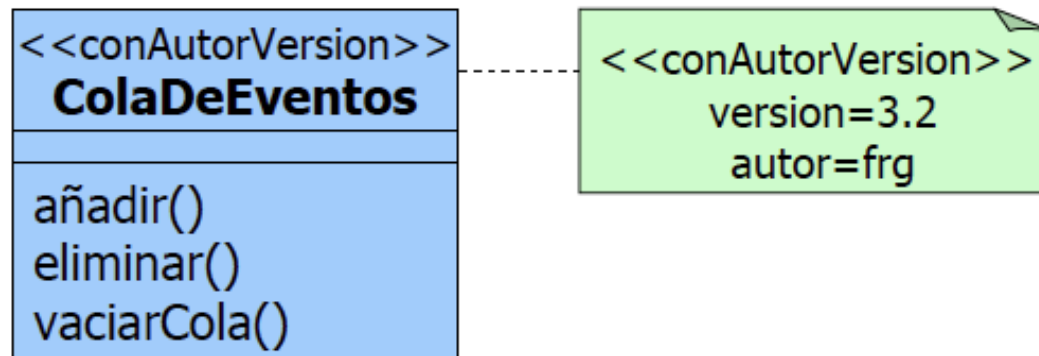


`<<exception>>`  
**Overflow**

# Mecanismos comunes - Extensibilidad

- Valor etiquetado

- Extiende las propiedades de un estereotipo de UML, permitiendo añadir nueva información en la especificación del estereotipo.
- Ejemplo: añadir *version* y *autor* mediante la creación del estereotipo `<<conAutorVersion>>` y asociándole una nota con los dos valores etiquetados.



# Mecanismos comunes - Extensibilidad

- **Restricción**

- Extiende la semántica de un bloque de construcción de UML, permitiendo **añadir nuevas reglas o modificar** las existentes.
- Ejemplo: restringir la clase *ColaDeEventos* para que todas las adiciones se hagan en orden.

