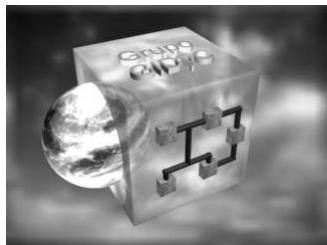


estudios de postgrado
en computación



Lenguaje Unificado de Modelado (UML)



Grupo de Ingeniería de Datos y
Conocimiento (GIDyC)

Departamento de Computación
Escuela de Ingeniería de Sistemas

Isabel Besembel Carrera



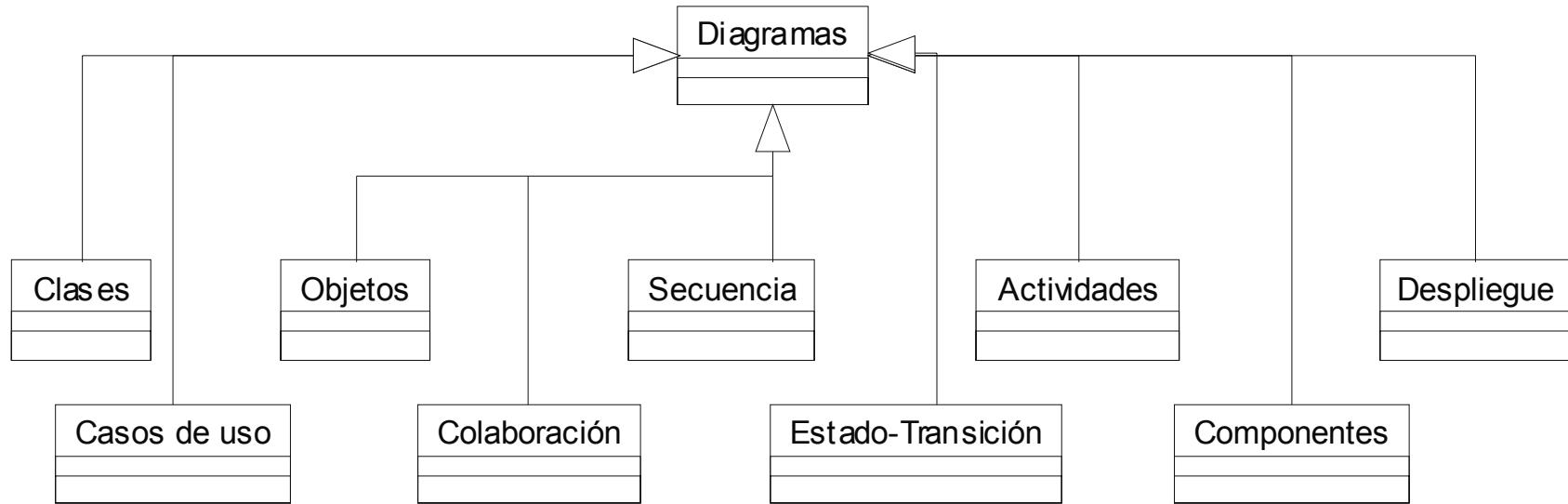
Génesis ...

- ☞ **Inicio de los ochenta:** Métodos orientados por objetos.
- ☞ **Finales de 1994:** nociones de clases y asociaciones (Rumbaugh-OMT), divisiones en subsistemas (Booch) y casos de uso (Jacobson)
- ☞ **1995:** Método unificado versión 0.8.
- ☞ **1996:** versión 0.9
- ☞ **1997:** UML versión 1.0

Génesis

- ☞ **Origen:** Fusión de las notaciones de Booch (1993), OMT (1996), OOSE y otras.
- ☞ **Características:** simplicidad, intuitiva, homogénea, coherente, genérica, extensible y configurable.
- ☞ **Objetivo:** Descripción de los artefactos de desarrollo de programas.
- ☞ **Contenido:** Nueve tipos de diagramas.

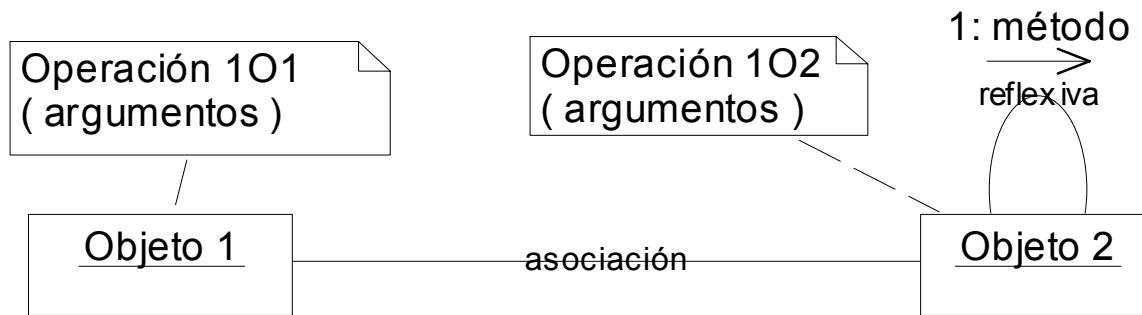
Diagramas



- ☞ El orden de la presentación no refleja el orden de implementación de un proyecto real.

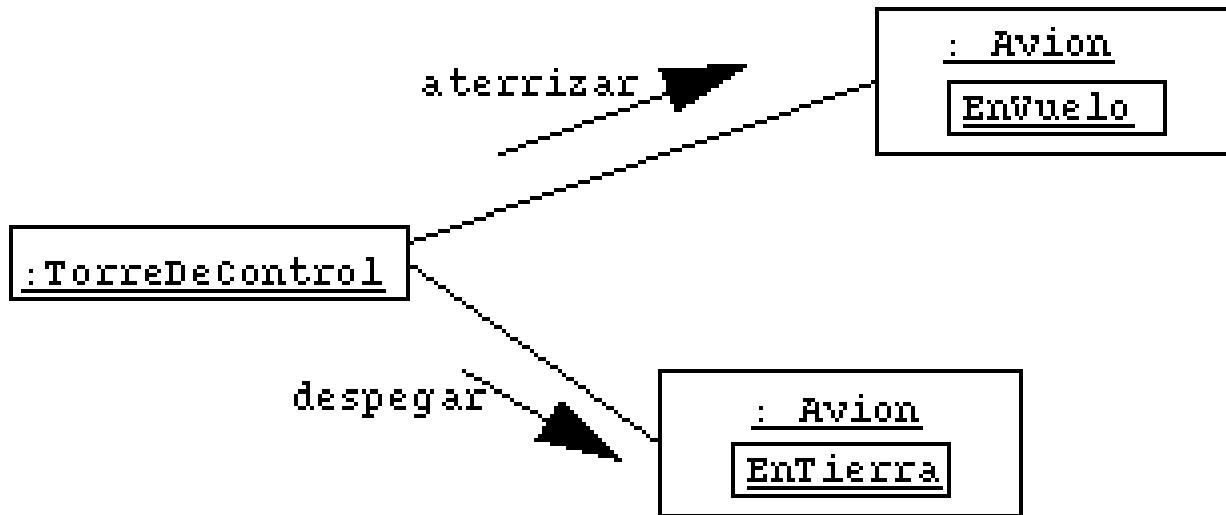
Conceptos básicos ...

- ☞ **Objeto:** representación de algo que se describe mediante un *identificador*, una *estructura* y un *comportamiento*.



Conceptos básicos ...

- ☞ **Atributos:** propiedades relevantes de un objeto que representa su estructura. Simples o compuestos.



Enlace entre comportamiento y atributos

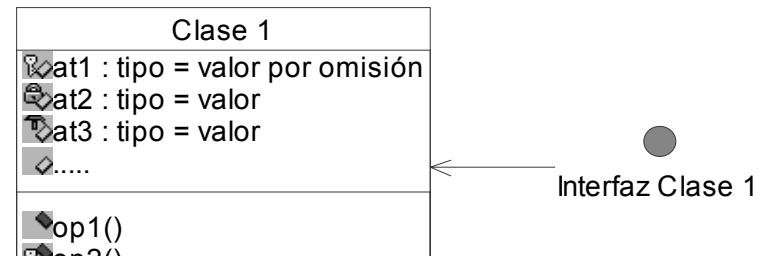
Conceptos básicos ...

- ☞ **Clasificación:** agrupación de objetos con propiedades y comportamiento similares dentro de una clase.
- ☞ **Clase:** objeto que define la estructura y el comportamiento de un conjunto de objetos que tienen el mismo patrón estructural y de comportamiento.
- ☞ **Instancia:** Cada objeto que pertenece a una clase.

Clase 1
at1 : tipo = valor por omisión
at2 : tipo = valor
at3 : tipo = valor
.....
op1()
op2()
op3()
.....()

Conceptos básicos ...

- ☞ **Extensión de clase:** conjunto de todas las instancias de una clase.
- ☞ **Instanciación:** proceso de generación o creación de las instancias de una clase.
- ☞ **Interfaz:** clase asociada que describe su comportamiento visible.

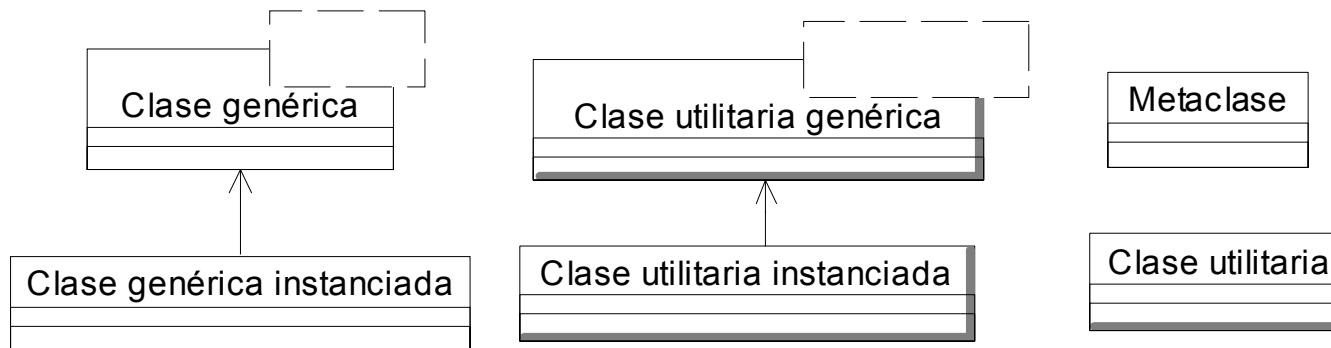


Clase 2

estructura

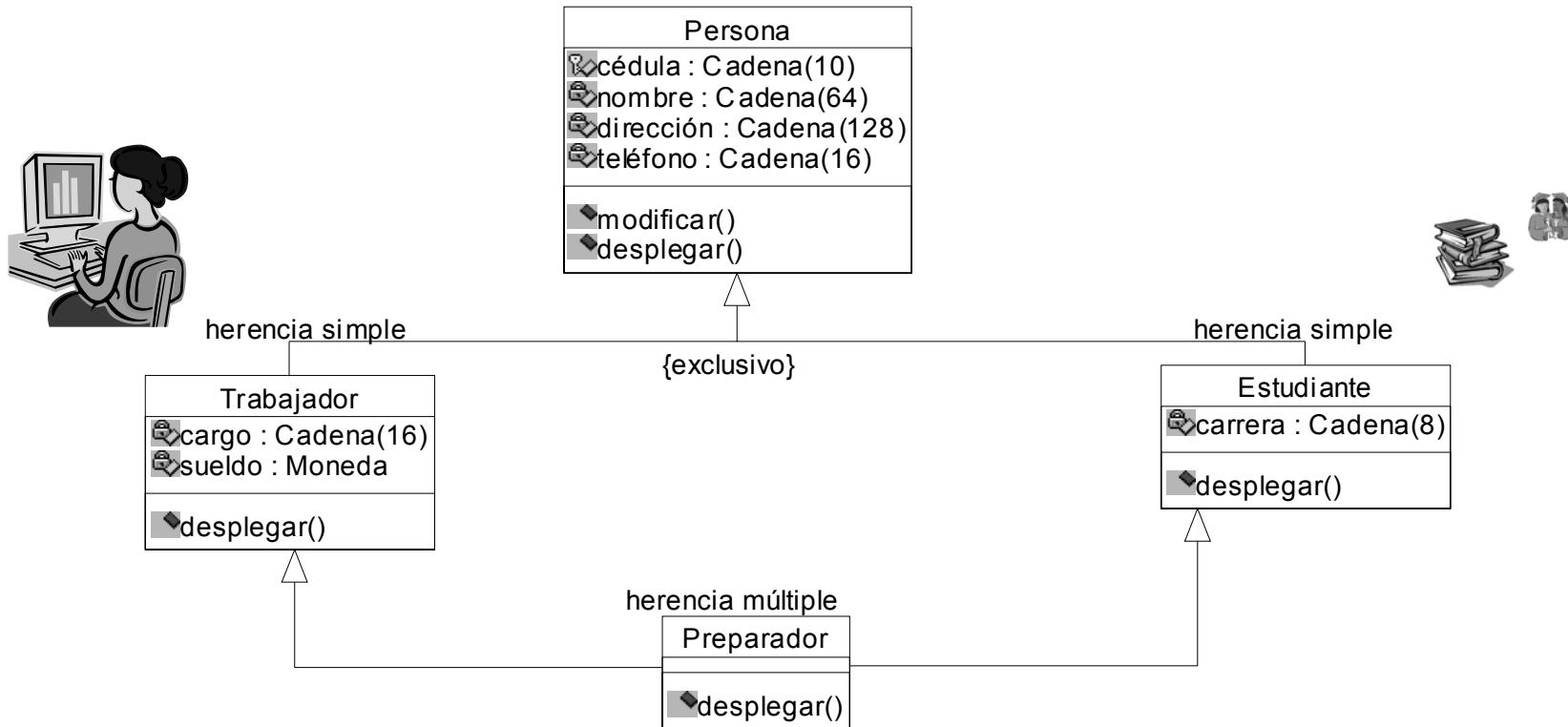
Conceptos básicos ...

- ☞ **Clases abstractas o parametrizables:** modelos de clases que se corresponden con clases genéricas. No son instanciables.
- ☞ **Clases utilitarias:** librerías de funciones.



Conceptos básicos ...

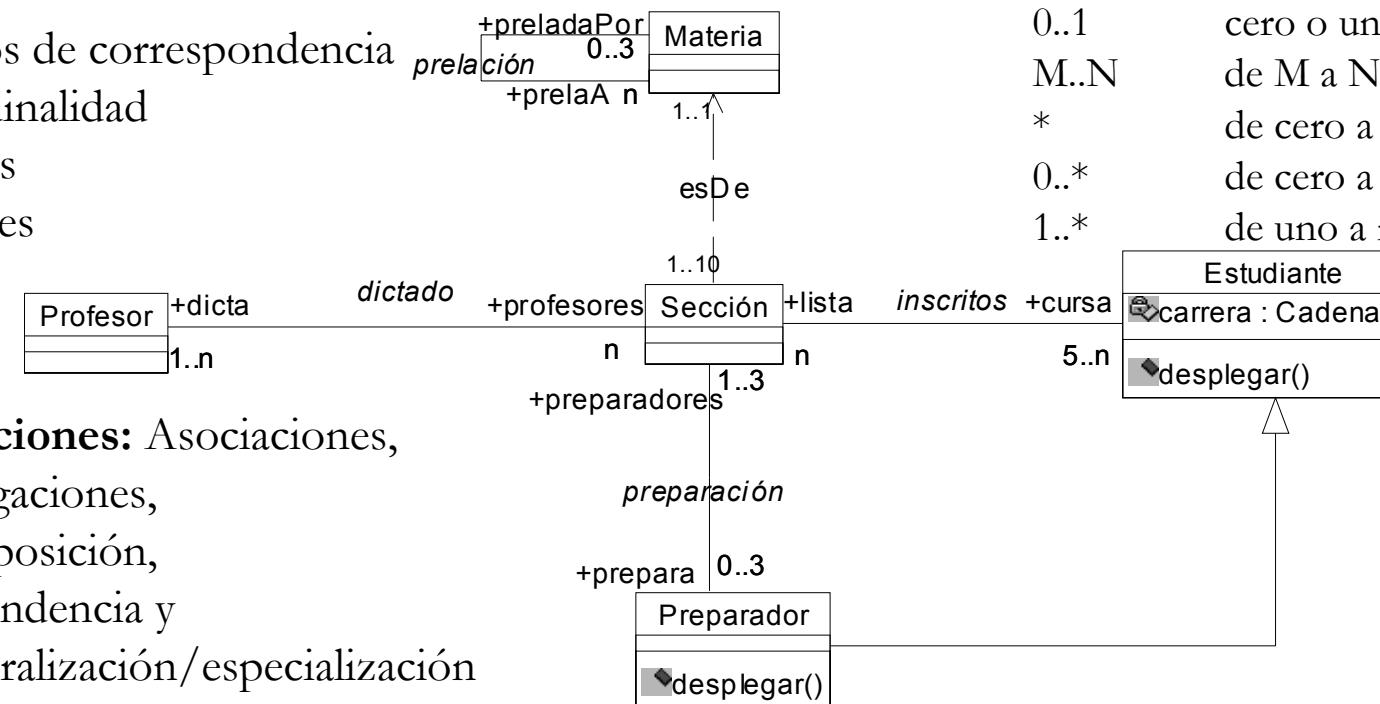
- ☞ **Jerarquía de clases:** relación ES-UN(A), abstracciones de generalización/especialización de clases.
- ☞ **Herencia:** propiedad que tienen las clases de heredar de sus superclases estructura y/o comportamiento. Simple o múltiple.



Conceptos básicos ...

- ☞ **Asociaciones:** relaciones estructurales entre las clases, pueden ser: reflexivas, binarias, etc.

Tipos de correspondencia



Relaciones: Asociaciones,
Agregaciones,
Composición,
Dependencia y
Generalización/especialización

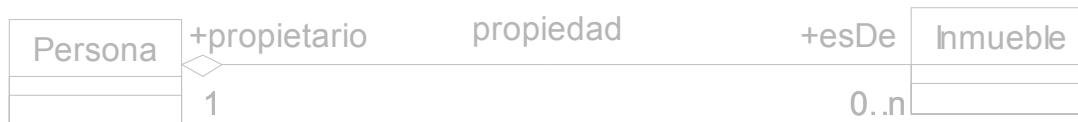
Multiplicidad de las asociaciones:

1	uno y sólo uno
0..1	cero o uno
M..N	de M a N (cardinales)
*	de cero a muchos
0..*	de cero a muchos
1..*	de uno a muchos

Conceptos básicos ...

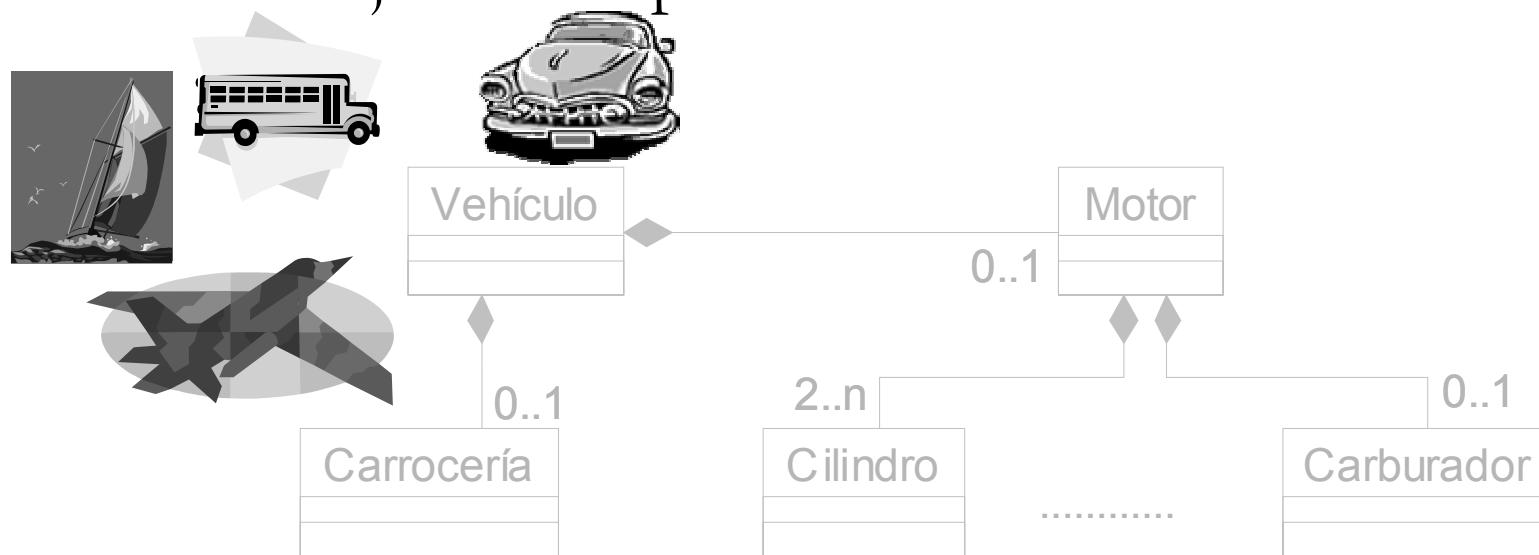
☞ Agregaciones:

- ☞ Una clase forma parte de otra
- ☞ Los valores de los atributos de una clase se propagan en los valores de los atributos de otra
- ☞ Una acción sobre una clase implica una acción sobre otra
- ☞ Los objetos de una clase están subordinados a los objetos de otra



Conceptos básicos ...

- ☞ **Composición:** relación ES-PARTE-DE, asimétrica, una de las extremidades juega un rol predominante en relación a la otra, por lo cual sólo lleva un rol. Indica clases de objetos compuestos.



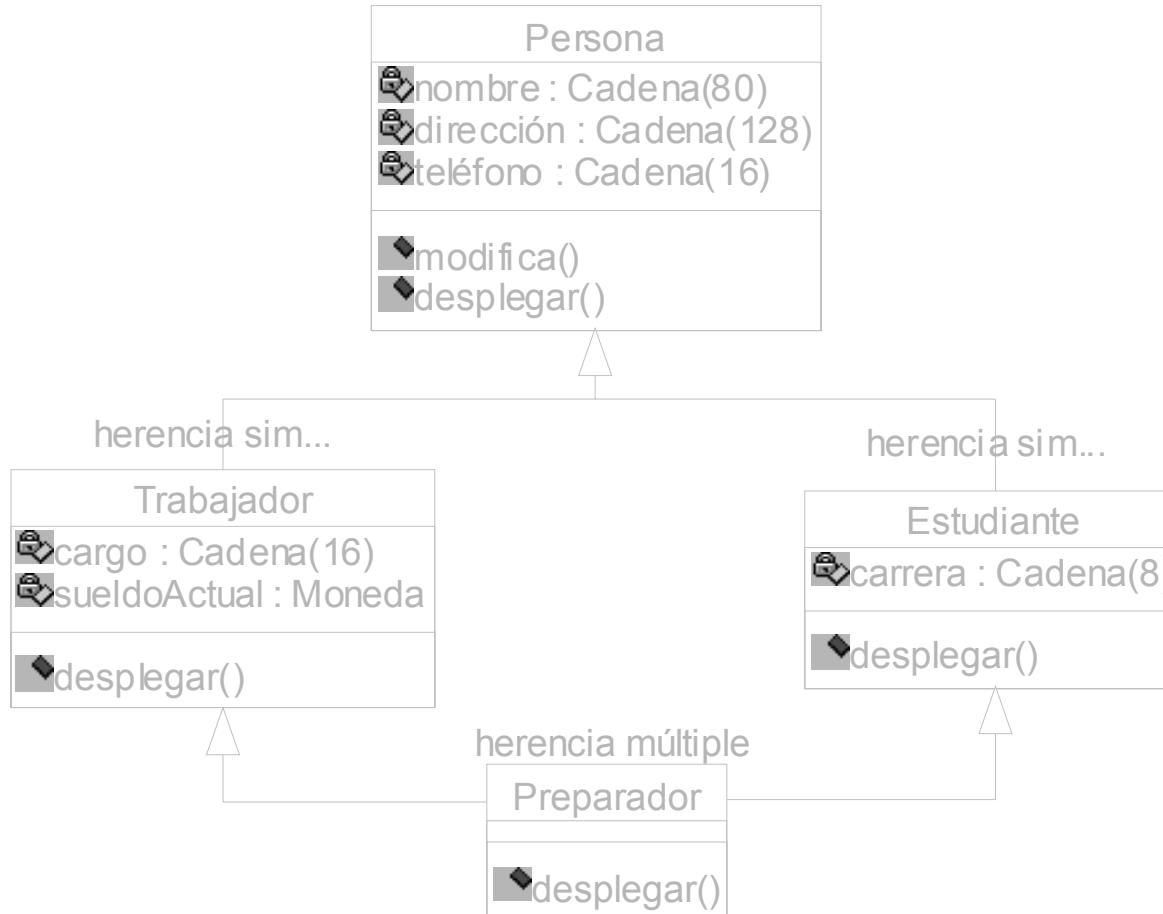
Conceptos básicos ...

- ☞ **Mensajes:** especificación de un objeto junto con la invocación de uno de sus métodos.
`objectIdDestino.nombreDelMétodo(listaDeArgumentos)`
- ☞ **Encapsulación:** propiedad que permite que los objetos sean definidos en su estructura y su comportamiento, obligando al uso del pase de mensajes o de la invocación de sus métodos, si se quiere acceder al objeto

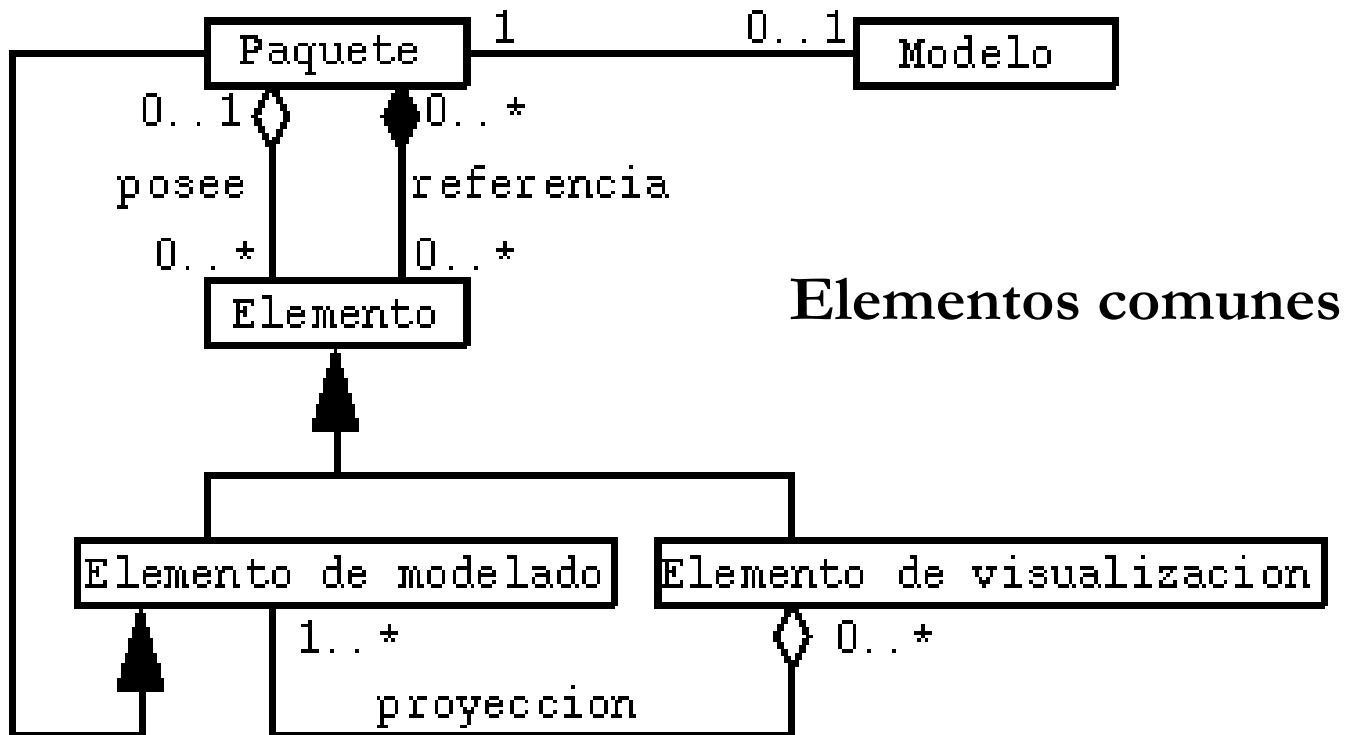
Conceptos básicos ...

- ☞ **Polimorfismo:** se puede usar el mismo nombre para la definición de un método en varias clases sin importar la relación entre las mismas.
- ☞ **Reescritura o sobrecarga:** permite nombrar código diferente con el mismo nombre para más de una clase de objetos.
- ☞ **Encadenamiento tardío:** permite seleccionar el código adecuado al objeto definido en la invocación del método.

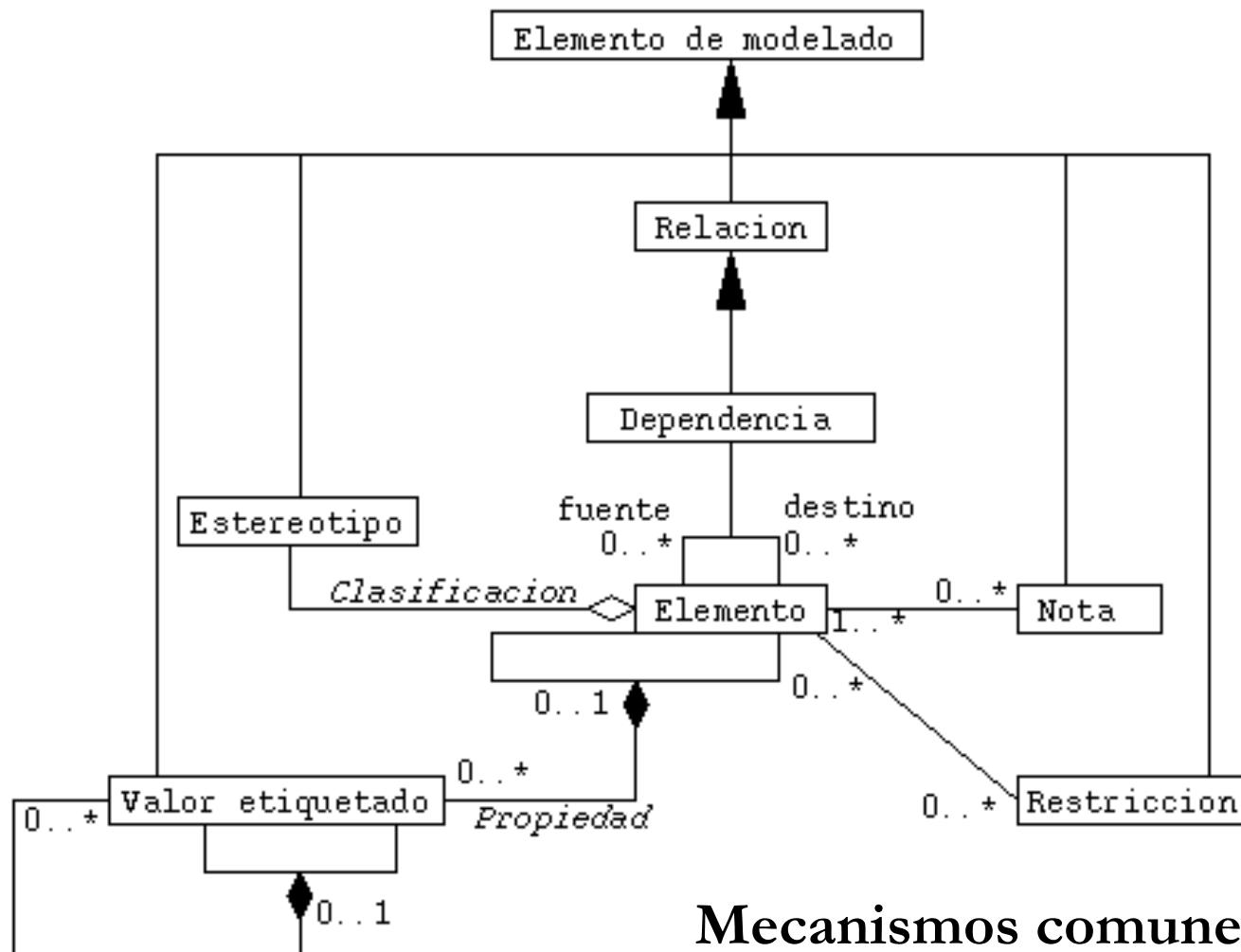
Ejemplo



Conceptos básicos (metamodelo)...

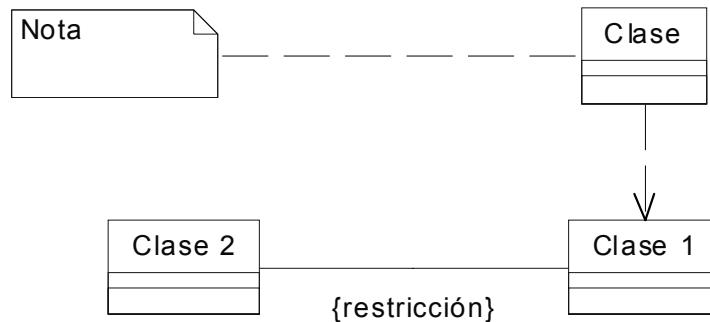


Conceptos básicos (metamodelo)



Conceptos básicos ...

- ☞ Estereotipos <<Semántica>>
- ☞ Etiquetas (nombre, valor)
- ☞ Notas
- ☞ Restricciones
- ☞ Dependencia
- ☞ Dicotomías:
 - ☞ (tipo, instancia) esencia y manifestación
 - ☞ (tipo, clase) especificación y realización



Tipos primitivos ...

- ☞ No son elementos de modelado \Rightarrow No poseen ni estereotipos, ni etiquetas, ni restricciones
- ☞ **Booleano:** tipo enumerado: Verdadero y falso
- ☞ **Expresión:** cadena de caracteres
- ☞ **Lista:** contenedor que puede ser ordenado e indizado
- ☞ **Multiplicidad:** conjunto no vacío de enteros positivos extendido con *

Tipos primitivos ...

- ☞ **Nombre:** cadena de caracteres que designa un elemento.

Nombre compuesto = nombre simple { '.'
nombre compuesto } .

Nombre calificado con el paquete al que pertenece = paquete ' :: ' nombre

- ☞ **Cadena:** cadena de caracteres con nombre

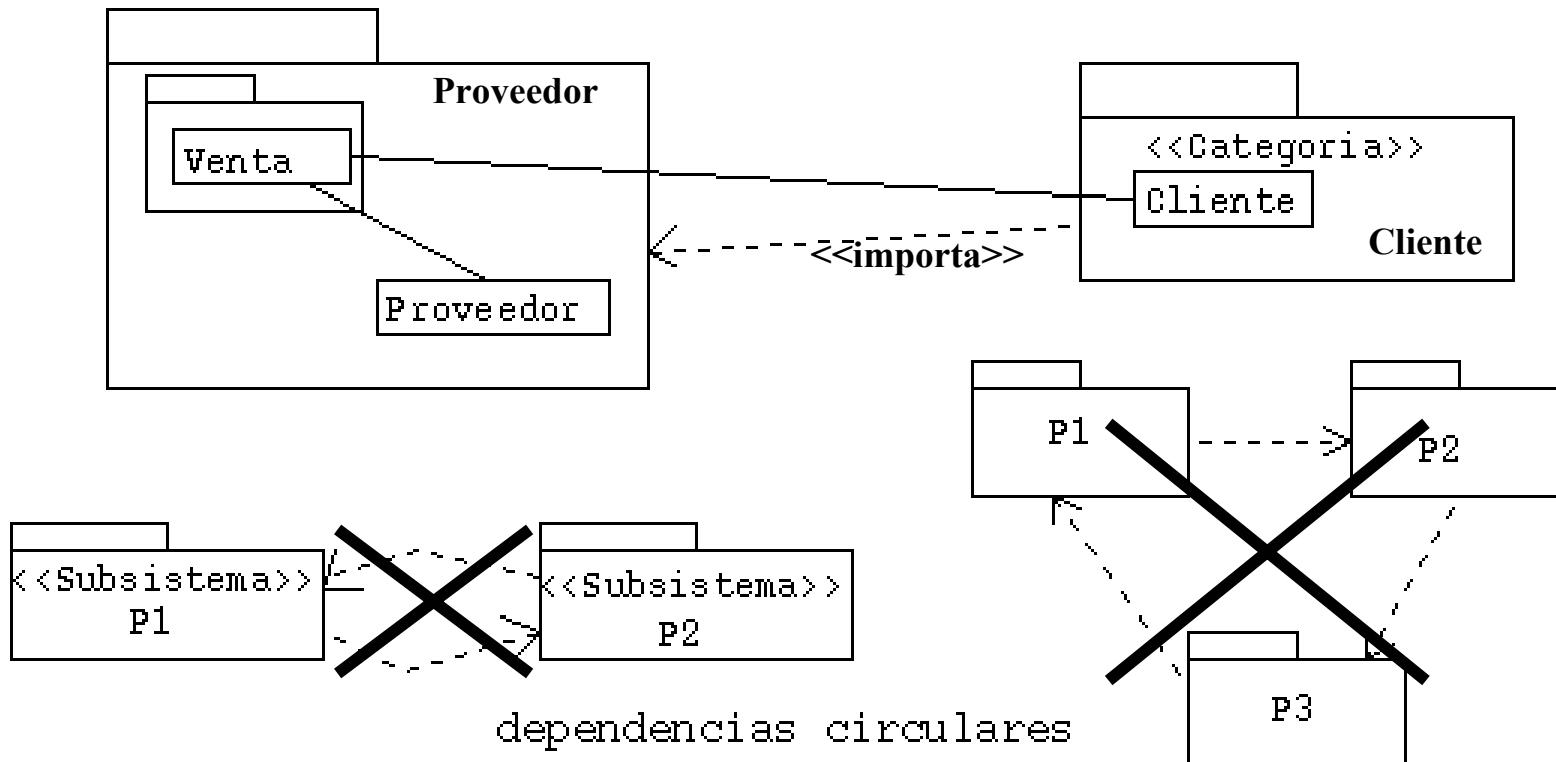
Tipos primitivos

- ☞ **Tiempo:** cadena que representa un tiempo absoluto o relativo
- ☞ **Punto:** tupla (x, y, z) posición en el espacio, por omisión $z = 0$
- ☞ **No interpretado:** blob, su significado depende del dominio

Paquetes ...

- ☞ Define un espacio de nombres. Dos paquetes diferentes pueden contener elementos con el mismo nombre.
- ☞ Puede contener otros paquetes y elementos de modelado, sin límite de anidamiento.
- ☞ Permiten dividir un modelo y reagrupar y encapsular los elementos de modelado.
- ☞ Poseen una interfaz y una realización.

Paquetes ...

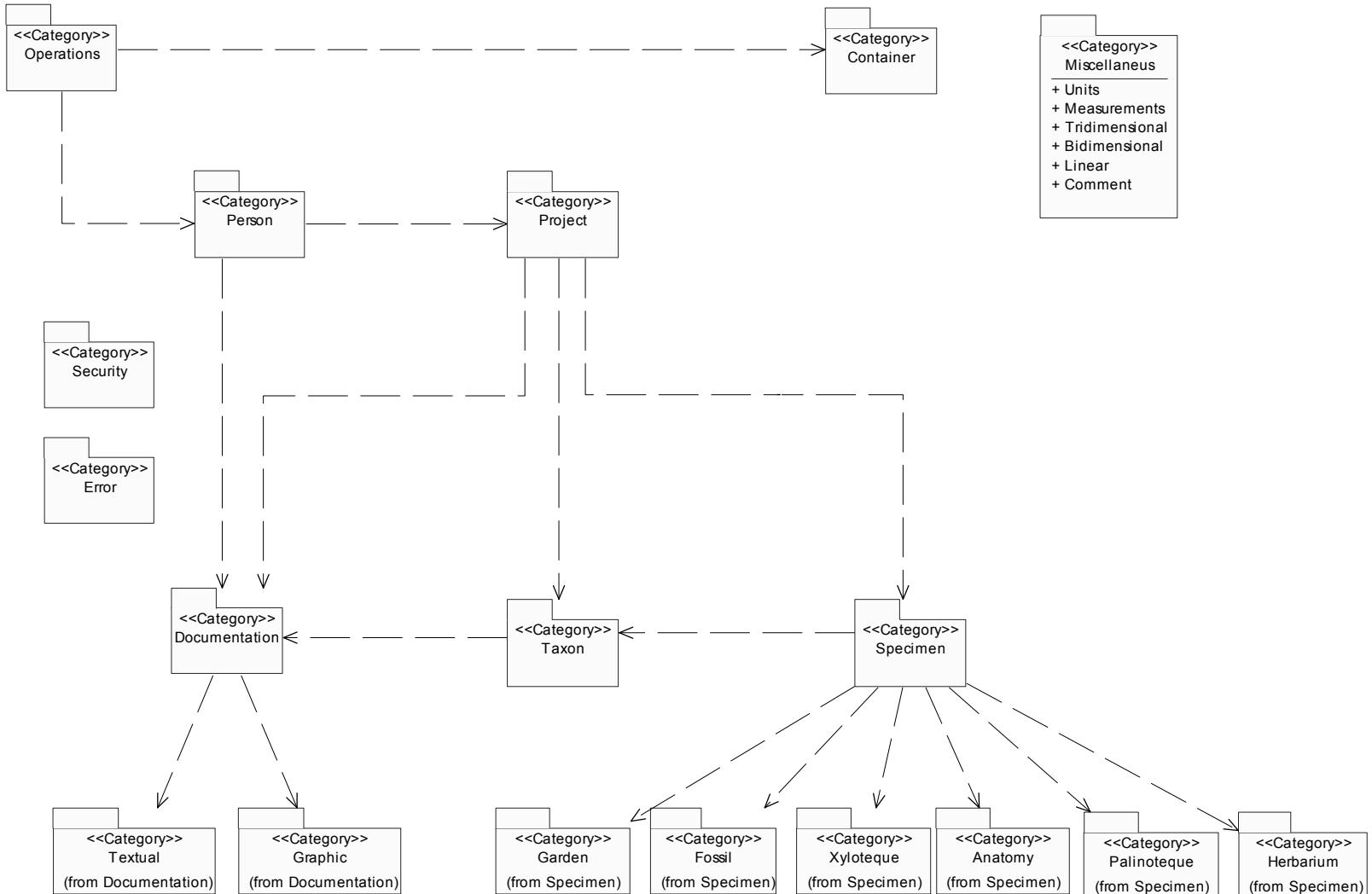


Paquetes ...

- ☞ Cada elemento indica si es visible o no (public o implementation).
- ☞ **Arquitectura del sistema:** se expresa con la jerarquía de paquetes y su red de relaciones de dependencia.
- ☞ El paquete de más alto nivel es el raíz.
- ☞ Una clase contenida en un paquete puede también aparecer en otro, bajo la forma de un elemento importado.

Paquetes (ejemplo)

Categorías de ORINOCO



Diagramas ...

- ☞ Permiten visualizar y manipular los elementos de modelado.
- ☞ Son grafos donde los nodos y los arcos tienen diferentes representaciones dependiendo del tipo de diagrama.
- ☞ Muestran todo o parte de las características de los elementos de modelado, según el nivel de detalle útil en el contexto del diagrama.

De clases ...

- ☞ Representa la estructura estática en términos de clases y asociaciones, las cuales pueden o no ir en paquetes y/o tener estereotipos (<<señal>>, <<interfaz>>, <<metaclase>> y <<utilidad>>).
- ☞ **Realización:**
 - ☞ *Listar los conceptos candidatos relacionados con los requerimientos considerados.*
 - ☞ *Dibujar el diagrama con ellos*

De clases ...

- ☞ *Anexar las asociaciones necesarias que se detecten.*
- ☞ *Anexar los atributos necesarios para cumplir los requerimientos.*
- ☞ Se recomienda hacer esto con el espíritu de un cartógrafo, es decir:
 - ☞ *Usar solamente los nombres existentes en el territorio.*
 - ☞ *Excluir las cosas irrelevantes.*
 - ☞ *No anexar cosas que **no** están allí.*

De clases ...

- ☞ Guía para identificar los conceptos
 - ☞ Si usted **no** puede pensar en el concepto *X* como un número, un texto o un valor atómico de la realidad, entonces *X* es probablemente un concepto y **no** un atributo.
 - ☞ Si tiene dudas, colóquelo como un concepto separado.
- ☞ Guía para los atributos
 - ☞ Tomar aquellos que son simples o puros (*Boolean, Date, numéricos, String, Color, etc.*)
 - ☞ Relacionar los conceptos con relaciones y **no** con atributos.

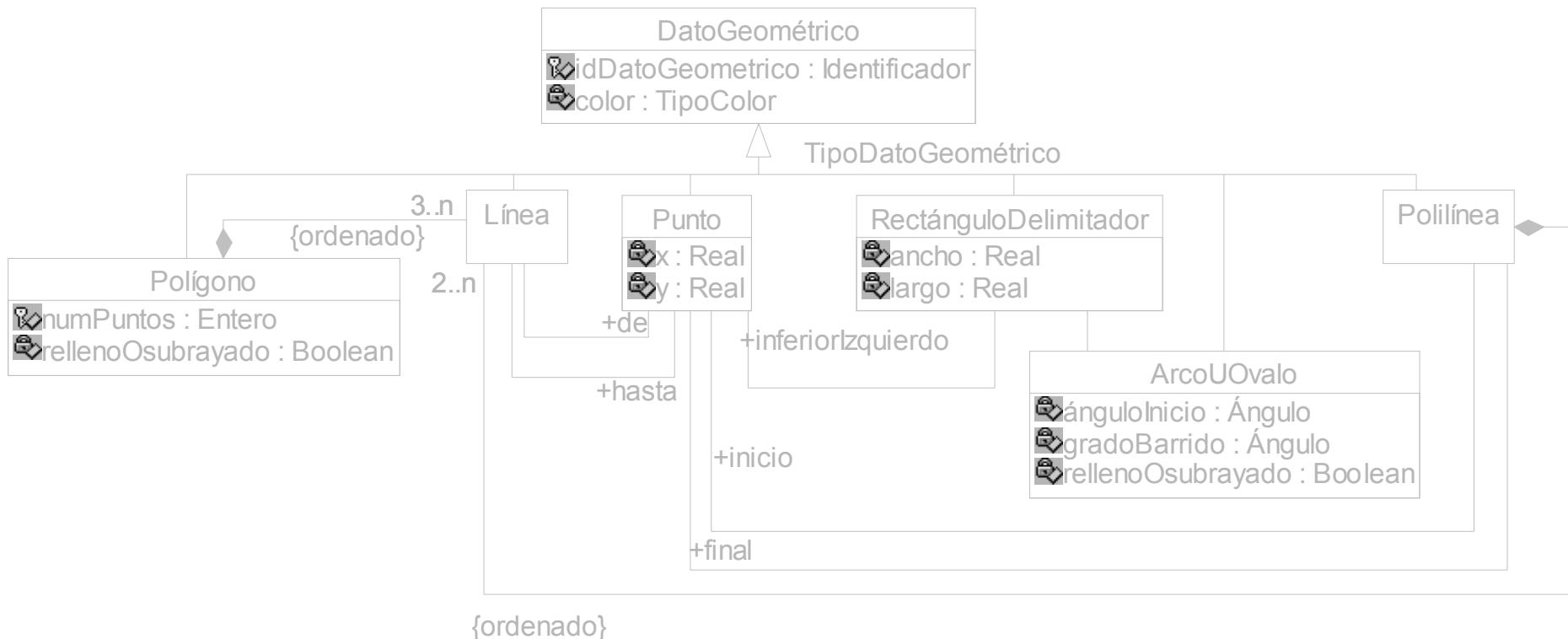
De clases ...

- ☞ Los tipos no primitivos son los tipos compuestos como por ejemplo: el valor de la cédula que está compuesto por una letra (E, P, V) más el número.
- ☞ Modelar los valores numéricos junto con su unidad de medida, ejemplo: moneda, velocidad, etc.
- ☞ Incluir un glosario de términos para mejorar la comunicación con el usuario (diccionario del diagrama).

De clases (ejemplo)



De clases (ejemplo)



De clases ...

☞ Navegación:

- ☞ La ausencia de flecha indica navegación en los dos sentidos

☞ Sintaxis:

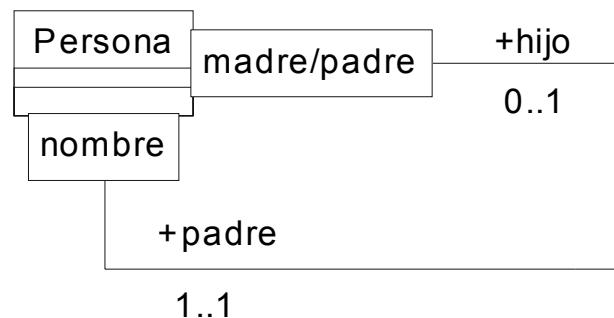
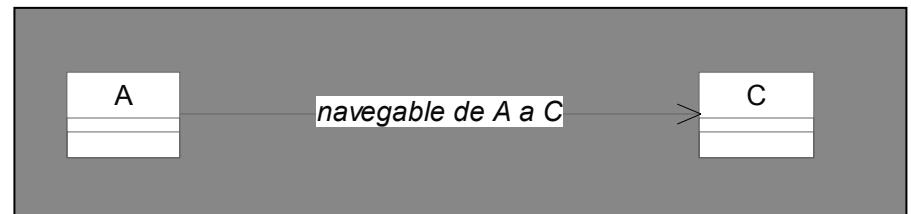
- ☞ destino ::= conjunto ‘?’ selector

destino: es un conjunto de valores u objetos

selector: nombre de atributo, nombre de asociación o nombre de función

Ejm: unaPersona.hijos

todos los hijos de una persona dada



De clases ...

☞ destino ::= conjunto ‘.’ ‘~’ selector

destino: es un conjunto de objetos obtenido por navegación en el orden inverso.

selector: nombre de función

Ejm: unaPersona.~hijos los padres de una persona

☞ destino ::= conjunto ‘[‘expresiónLógica’]’

Ejm: unaPersona.hijos[edad >=18]

☞ destino ::=conjunto ‘.’ selector ‘[‘ valorDeClave ’]’

Ejm: unaPersona.hijos[unNombre]

De clases ...

☞ Relaciones:

- ☞ Guía para encontrar las relaciones
 - ☞ Considerar aquellas cuyo conocimiento necesita ser guardado o preservado durante un tiempo
 - ☞ Es más importante identificar los conceptos que identificar las relaciones
 - ☞ Muchas relaciones hacen confuso el diagrama
 - ☞ Evitar tener relaciones redundantes o derivables
 - ☞ Nombrarlas utilizando el formato: nombreTipo - frase verbal – nombreTipo

De clases ...

☞ Ejemplo de relaciones:

A es parte física de B (MI)

A va después de B

A usa o maneja B

A está contenida físicamente en B (MI)

A está relacionada con una transacción B

A es una subunidad organizacional de B

A es una transacción relacionada con otra B

A es conocida, registrada o capturada en B (MI)

A es miembro de B

A es parte lógica de B (MI)

A es propietaria de B

A es un ítem de una transacción o reporte B

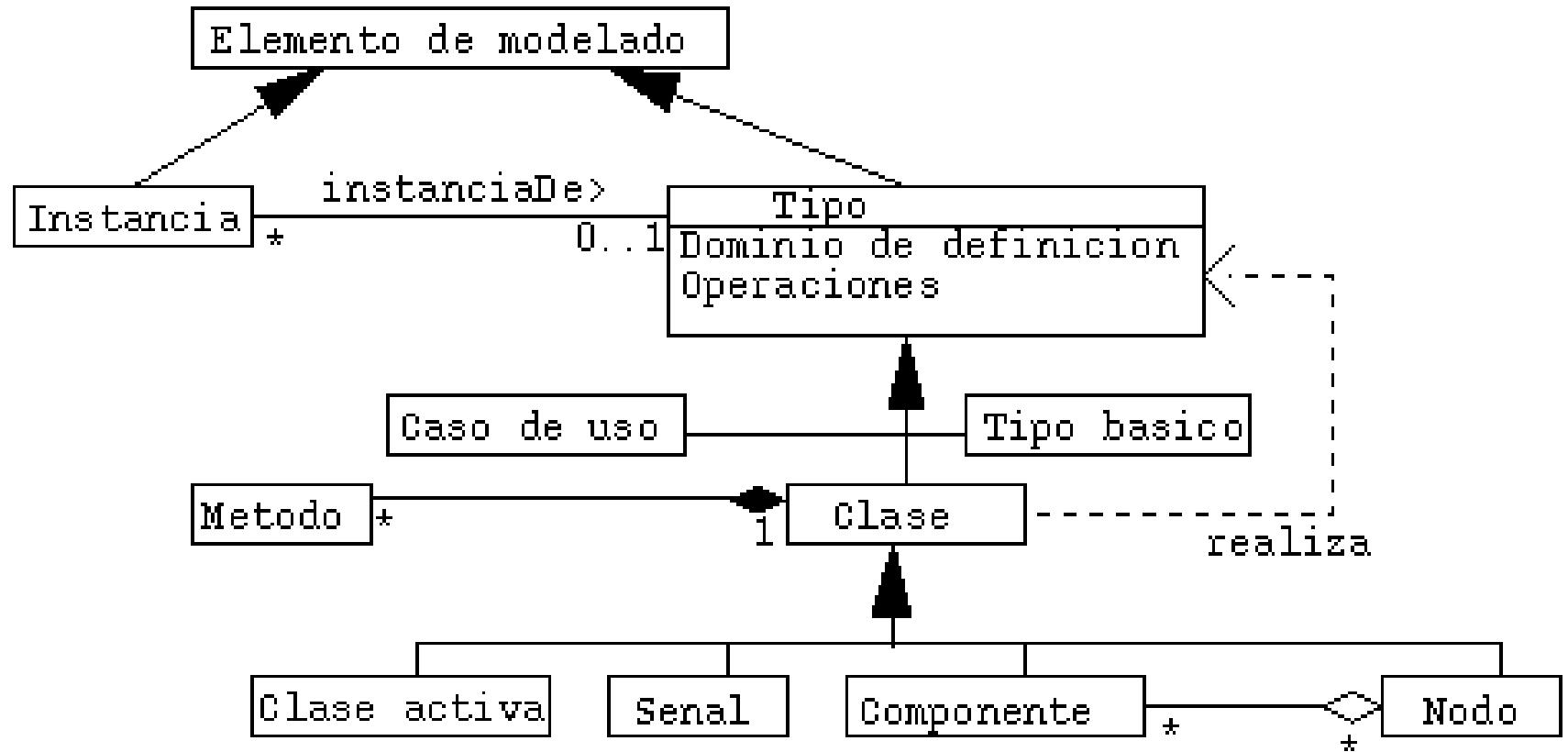
A está contenida lógicamente en B (MI)

A es una descripción de B

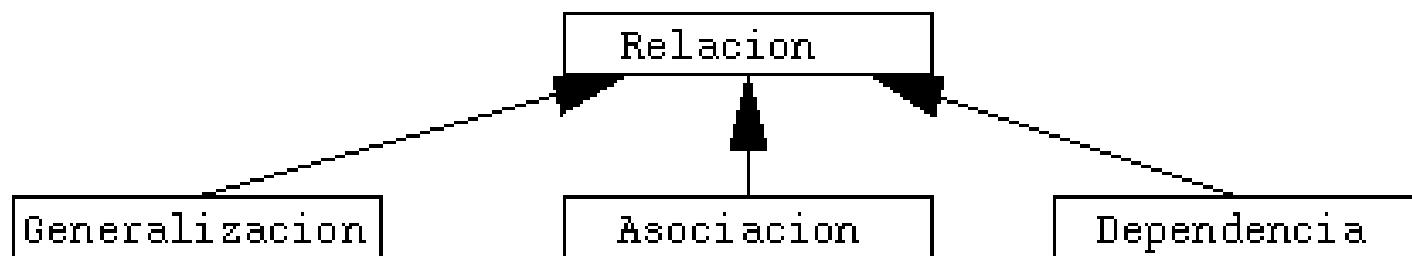
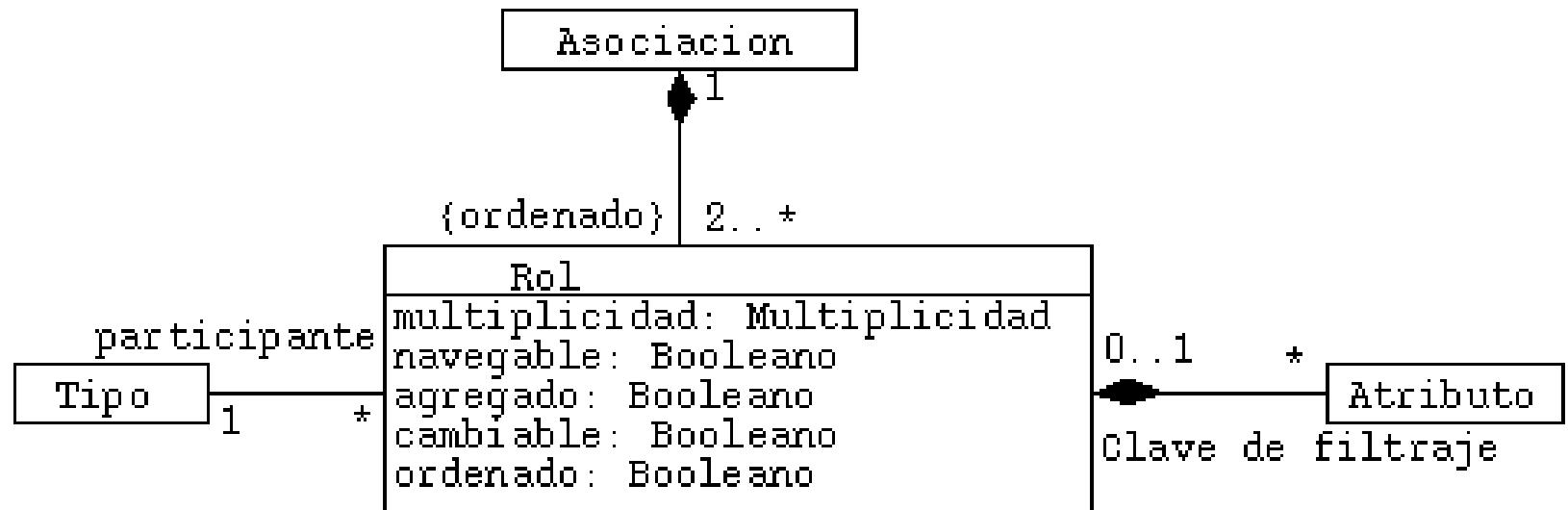
A se comunica con B

(MI: muy importantes considerarlas)

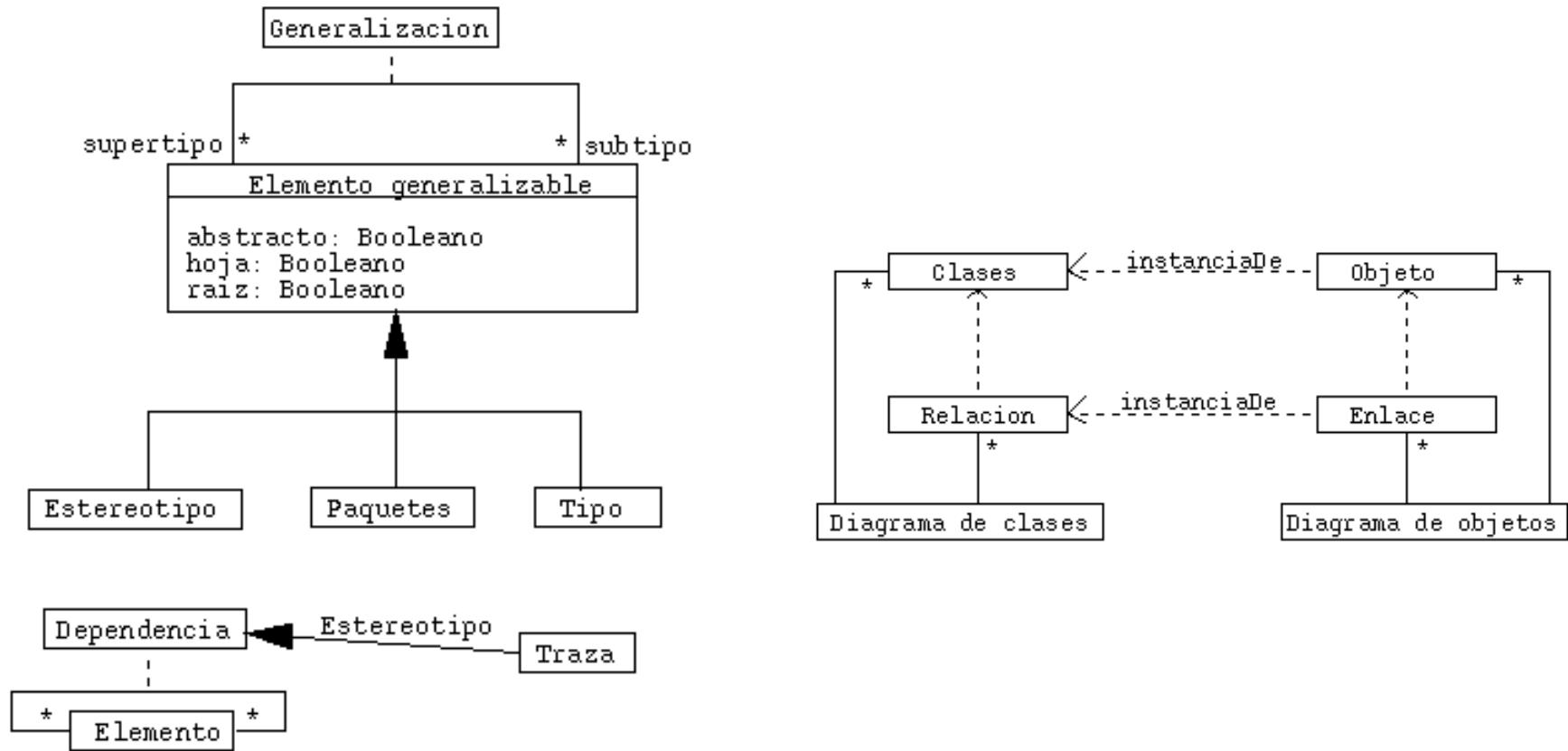
Metamodelo ...



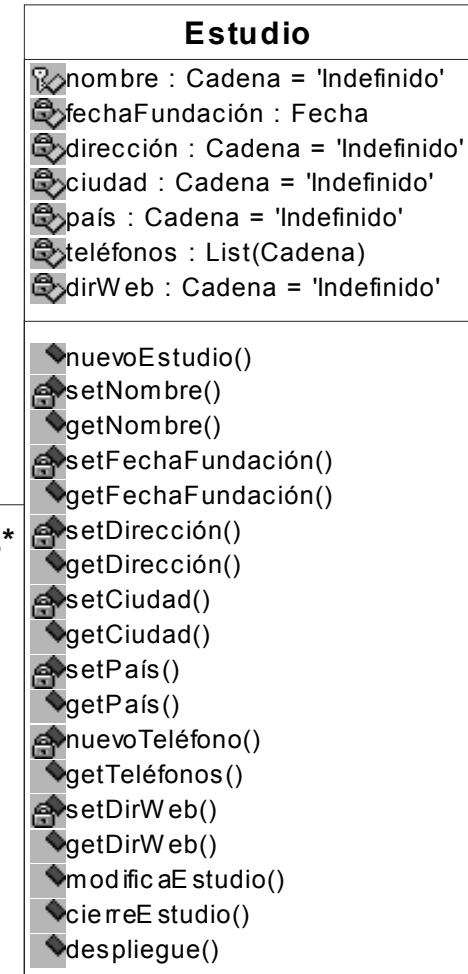
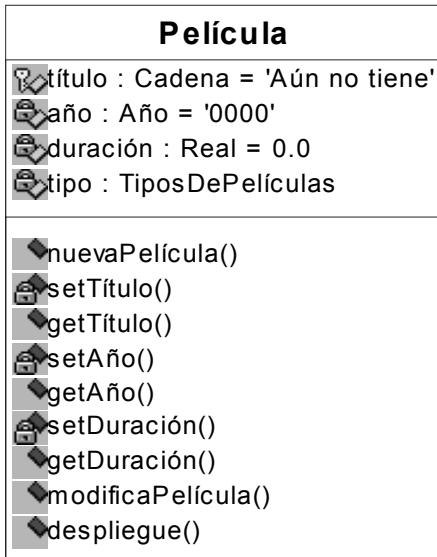
Metamodelo ...



Metamodelo ...



De clases (ejemplo)



Diagramas de casos de uso ...

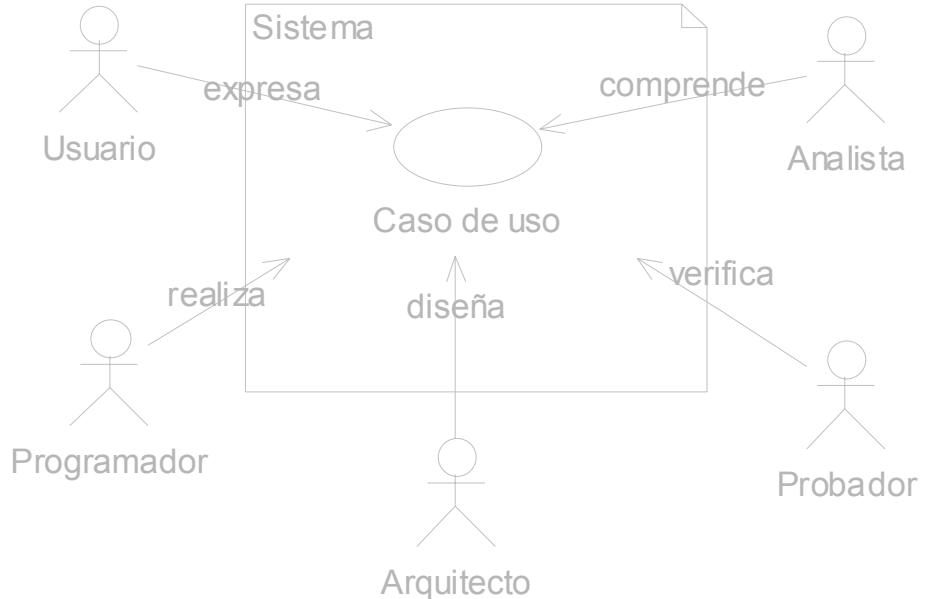
- ☞ Es una descripción de un proceso fin-a-fin relativamente largo que típicamente incluye varias etapas o transacciones, **no** es una etapa o actividad individual de un proceso.
- ☞ Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario; permiten definir los límites del sistema y las relaciones entre el sistema y su entorno.
- ☞ *Antes de hacerlos, se debe tratar de entender los requerimientos del sistema, expresando lo que el sistema debe hacer, a saber: El sistema debe hacer 1, 2, 3, ..., etc.*
- ☞ *Por cada función (1, ..., n) numerada, se coloca su descripción y una categoría ∈ {evidente | escondida | opcional}.*

Casos de uso ...

- ☞ Es una manera específica de utilizar el sistema.
- ☞ **Descripción:**
 - ☞ **Nombre:** <nombreDelCasoDeUso>
 - ☞ **Actores:** <listaDeActores> indicando quien lo inicia o dispara
 - ☞ **Propósito:** Intención del caso de uso
 - ☞ **Descripción:** Descripción de lo que hace el caso de uso
 - ☞ **Tipo:** {primario | secundario | opcional}, {abstracto | concreto}
 - ☞ **primario** *indica si es un proceso principal o importante,*
 - ☞ **secundario** *si es un proceso raro o de menor importancia,*
 - ☞ **opcional** *si sería deseable tener dicho proceso*
 - ☞ **Referencias cruzadas:** casos de uso relacionados y funciones.

Casos de uso ...

- ☞ Es la imagen de una funcionalidad del sistema, desencadenada en respuesta a la estimulación de un **actor** externo.
- ☞ Un **actor** representa el *rol* jugado por una persona o cosa que interactúa con el sistema. La misma persona puede jugar varios roles y varias personas pueden jugar el mismo rol



Casos de uso ...

☞ Categorías de actores:

- ☞ **Principales:** las personas que utilizan las funciones principales.
- ☞ **Secundarios:** las personas que efectúan tareas administrativas o de mantenimiento.
- ☞ **Externos:** los dispositivos que forman parte del dominio de aplicación y que deben ser utilizados.
- ☞ **Otros sistemas:** los sistemas con los que debe interactuar.
- ☞ Los casos de uso se determinan observando y precisando, actor por actor, las secuencias de interacción (escenarios) desde el punto de vista del usuario.

Casos de uso ...

☞ Relaciones entre los casos de uso:

- ☞ <<**dispara**>>, el actor inicia la interacción indicado con una flecha →
- ☞ <<**usa**>>, una instancia del caso de uso fuente utiliza el comportamiento del caso de uso destino →
- ☞ <<**extiende**>>, un caso de uso extiende el caso de uso destino. →
- ☞ Sólo hay un actor por cada caso de uso.
- ☞ Para su realización es conveniente responder las preguntas:
 - ☞ ¿Cuáles son las tareas del actor?
 - ☞ ¿Cuáles datos o información el actor debe crear, guardar, modificar, destruir o leer?
 - ☞ ¿Debe el actor informar al sistema de los cambios externos?
 - ☞ ¿Debe el sistema informar al actor las condiciones internas?.

Casos de uso ...

- ☞ La descripción se debe concentrar en lo que DEBE HACERSE y NO en la manera de hacerlo.
- ☞ Si un caso de uso es muy complejo (por ejemplo más de diez páginas) es posible dividirlo en casos más pequeños.
- ☞ El enfoque orientado por objetos materializa un caso de uso por medio de la colaboración entre los objetos.
- ☞ Los escenarios se representan con los diagramas de interacción.

Casos de uso ...

- ☞ La expansión de los casos de uso contienen la descripción de las acciones del actor y la respuesta del sistema, colocados en dos columnas.

Acción del actor	Respuesta del sistema
1) Se inicia cuando el actor inicia un evento i) n) Termina cuando j)

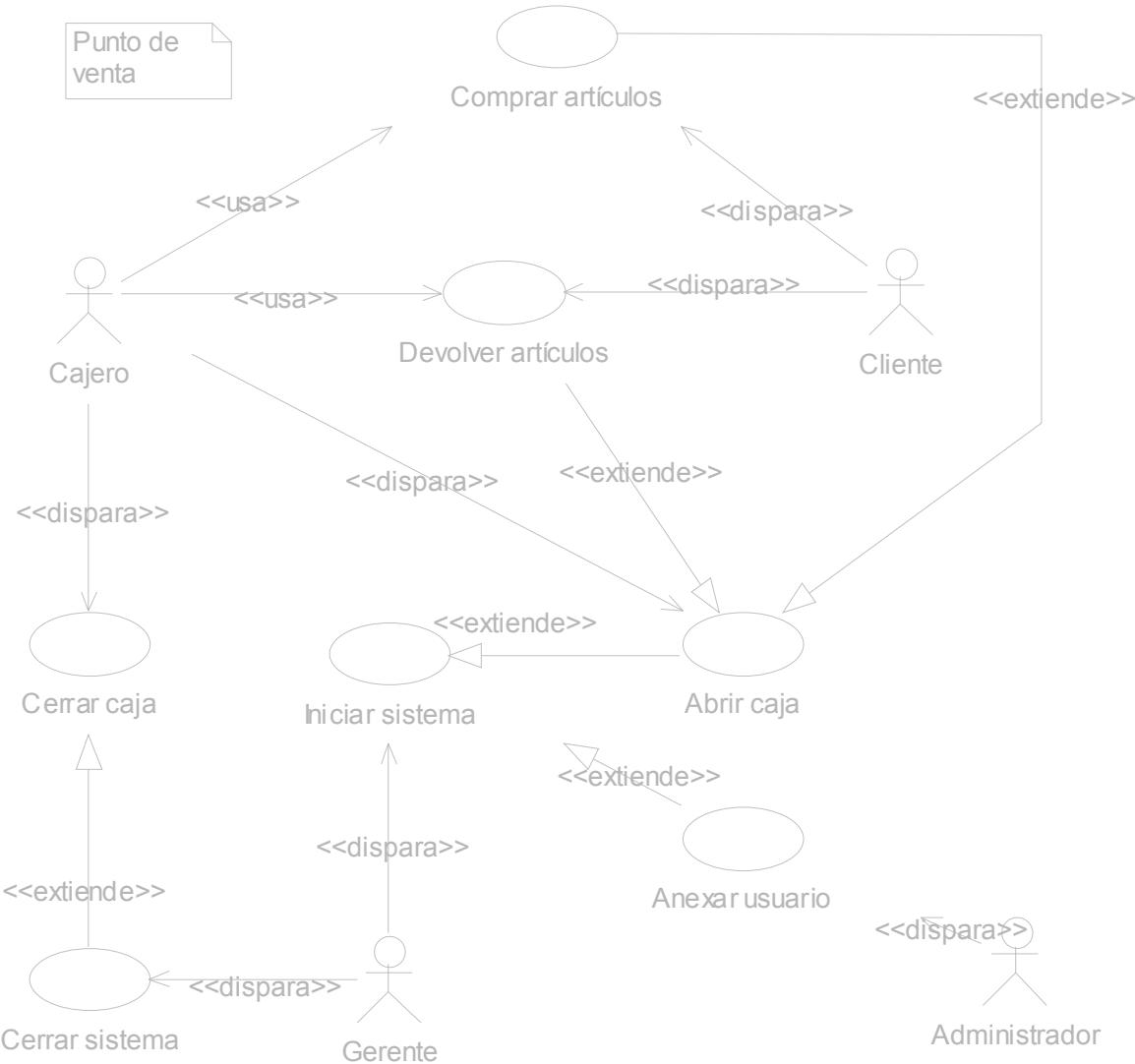
Casos de uso ...

- ☞ Los nombres de los casos de uso se inician con un verbo y se asocian a cada actor que lo dispara.

Actor	Caso de uso
Cajero	Abrir caja Cerrar caja
Cliente	Comprar artículos Devolver artículos
Gerente	Iniciar sistema Cerrar sistema
Administrador del sistema	Incluir nuevo usuario



Casos de uso ...



Casos de uso ...

- ☞ Se pueden **ordenar por prioridades** según los factores propuestos por Larman, a saber:
 - a. Impacto significativo sobre la arquitectura de diseño, como: anexa muchas clases al dominio o requiere servicios de persistencia.
 - b. Información significativa se obtiene con relativamente poco esfuerzo.
 - c. Incluye riesgos, funciones complejas o críticas en tiempo.
 - d. Involucra investigación significativa o nueva y riesgosa tecnología.
 - e. Representa una línea de negocios primaria.
 - f. Directamente soporta un incremento de beneficios o un decrecimiento de los costos

Casos de uso

- ☞ Se coloca una escala numérica para cada factor y se realiza una tabla.

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>total</i>
Abrir caja	4	4	1	0	5	3	17
Cerrar caja	4	4	3	0	5	3	19
Comprar artículos	5	3	2	0	5	3	18

Luego, en base a la puntuación se clasifican por prioridad en: alta, media y baja, colocando una justificación de la misma.

Prioridad	Caso de uso	Justificación
Alta	Abrir caja	Permite un incremento de los beneficios de la empresa
Alta	Comprar artículo	Incrementa los beneficios de la empresa
Alta	Cerrar caja	Constata los beneficios de la empresa

Escenarios

- ☞ Por cada caso de uso del sistema:
 - ☞ Definir los escenarios principales
 - ☞ Por cada escenario principal:
 - ☞ Definir su diagrama de interacción

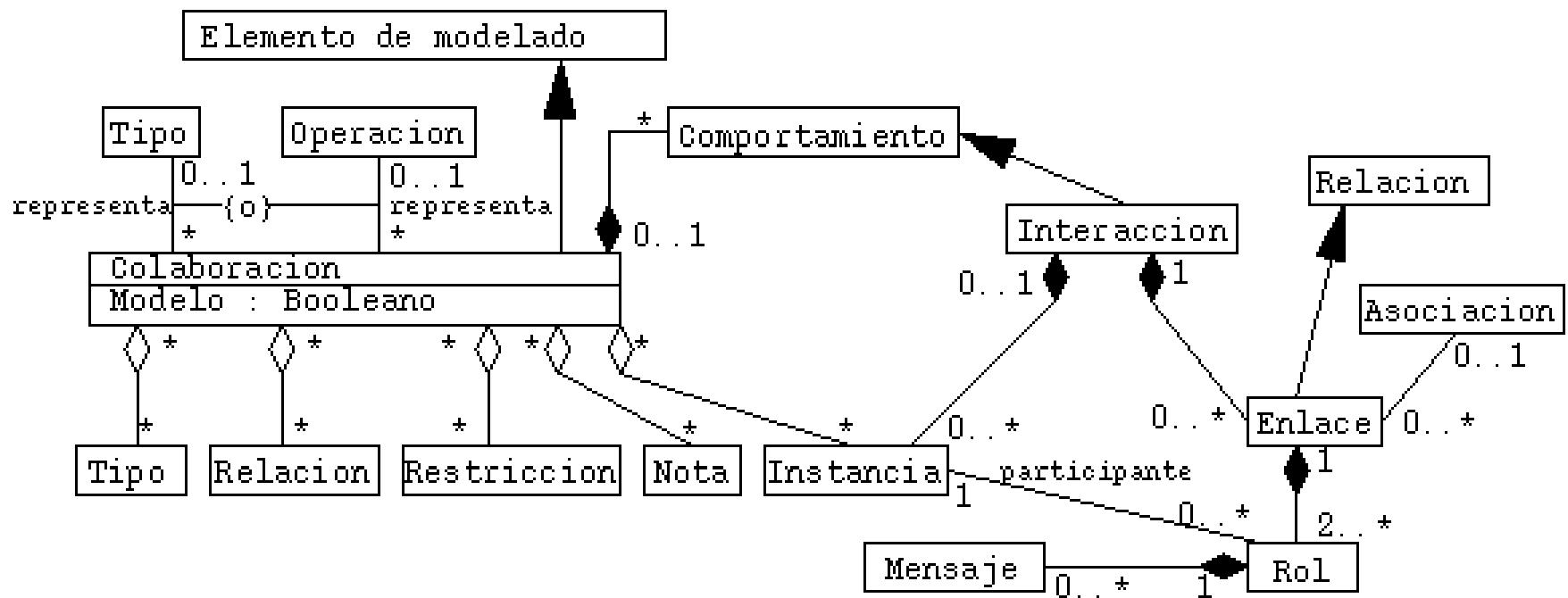
Caso de uso	Escenarios
Iniciar sistema	Identificación
Iniciar sistema	Iniciar cajas
Abrir caja	Identificación
Abrir caja	Definir condiciones iniciales
Comprar artículos	Identificación del artículo
Comprar artículos	Modificación del inventario

Diagramas de interacción

- ☞ Visualizada desde el punto de vista del tiempo (diagramas de secuencia) o del espacio (diagramas de colaboración).
- ☞ **Diagrama:**
 - ☞ De secuencias: Muestran las interacciones entre los objetos desde el punto de vista temporal insistiendo en la cronología del envío de mensajes.
 - ☞ De colaboración: Expresan el contexto de un grupo de objetos (los objetos y sus enlaces) y la interacción entre ellos (envío de mensajes entre ellos), son una extensión de los diagramas de objetos.



De interacción (metamodelo)



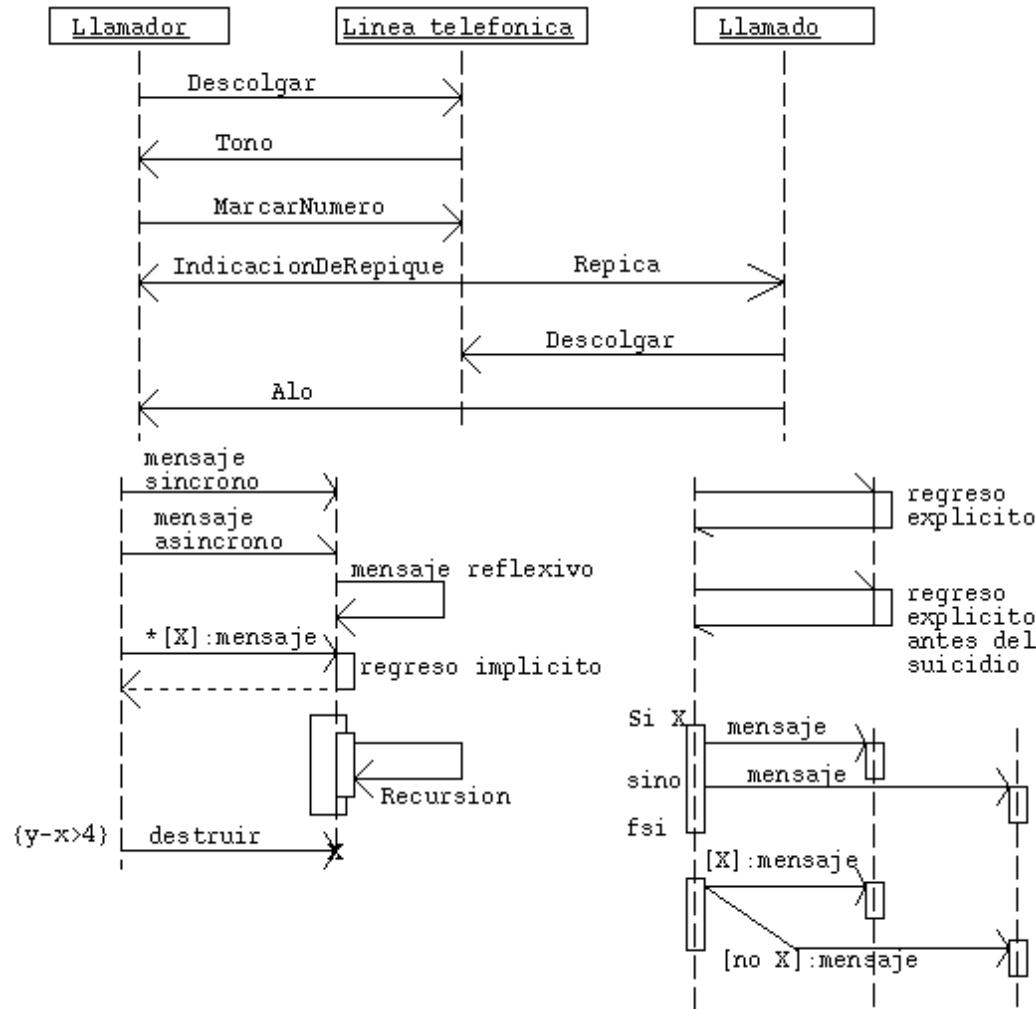
De secuencias ...

- ☞ Los objetos se comunican intercambiando mensajes representados por flechas horizontales orientadas desde el objeto fuente al destinatario.
- ☞ Los envíos de mensajes pueden ser asíncronos (la fuente no espera que el destinatario trate el mensaje y continua con sus quehaceres) y sincrónicos (la fuente espera que el destinatario termine de tratar su mensaje).
- ☞ Los mensajes pueden ser reflexivos, recursivos, condicionados, e iterativos.

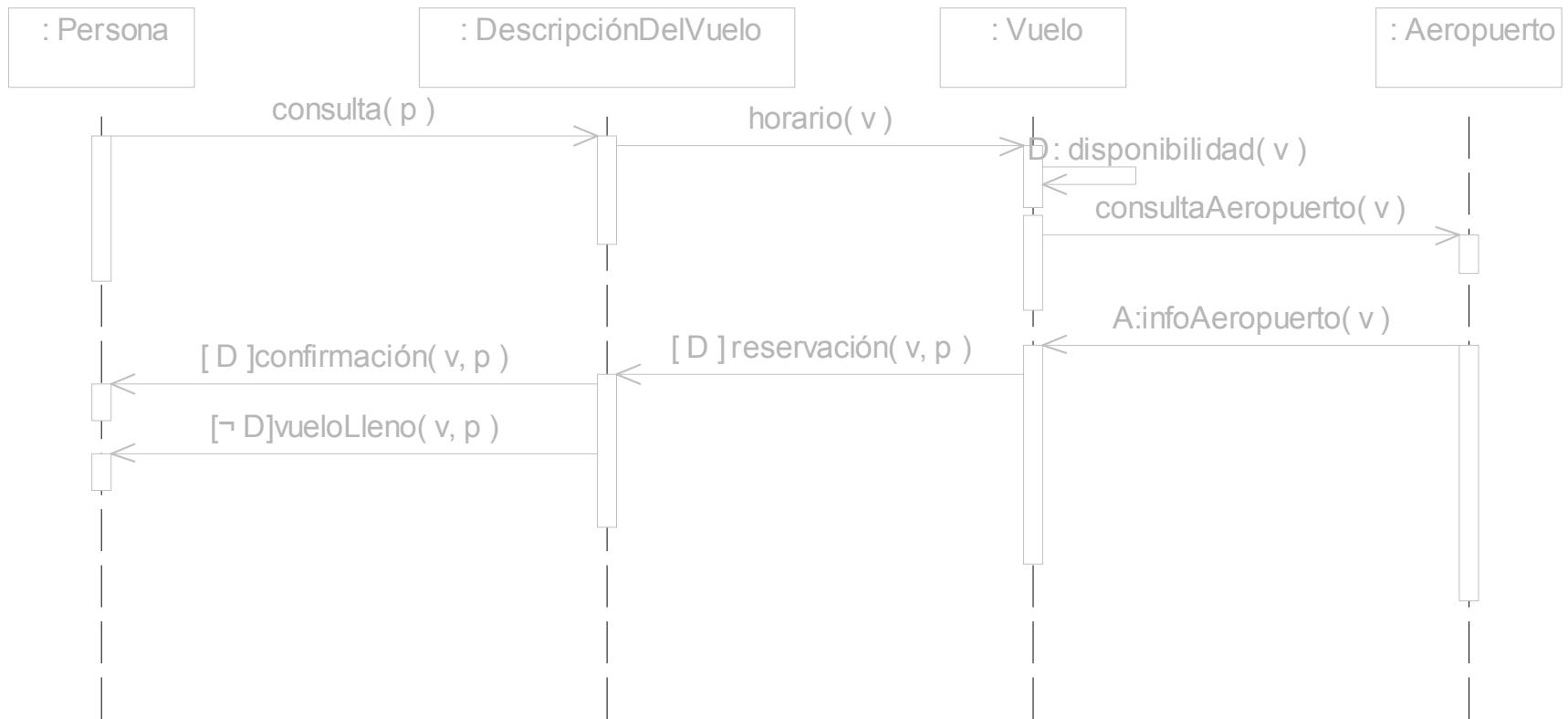
De secuencias ...

- ☞ Los objetos y los enlaces creados o destruidos en el curso de una interacción pueden llevar las restricciones de: {nuevo}, {destruido}, {transitorio}, iteración sobre todos los objetos de un tipo: *{todos}.
- ☞ **Forma del mensaje:**
sincronización secuencia : resultado := mensaje(argumentos)
- ☞ Se pueden expresar en la parte de secuencia las iteraciones:
 - ☞ *[i:=1..n] o
 - ☞ *[X<Y],
 - ☞ *paralelismo* *|||,
 - ☞ *condiciones* [A>B+1].

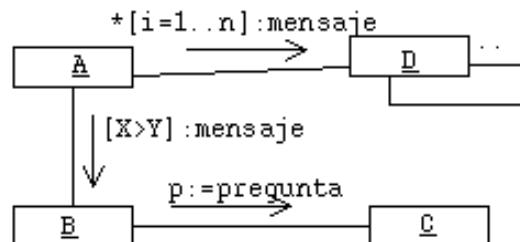
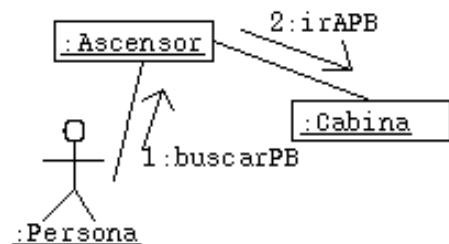
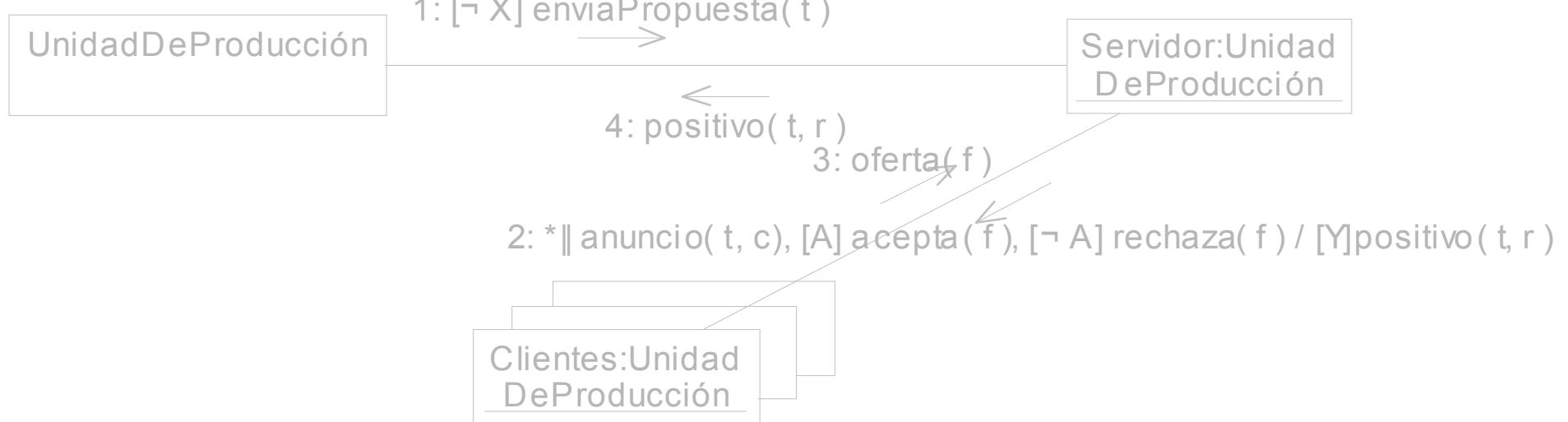
De secuencias ...



De secuencias



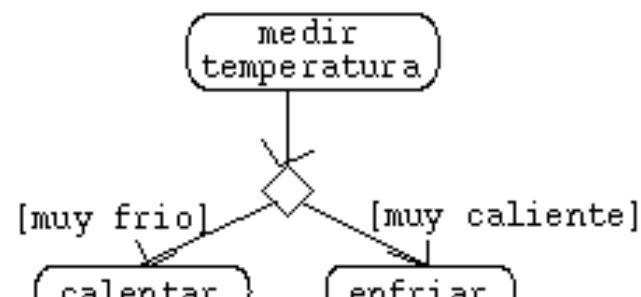
De colaboración



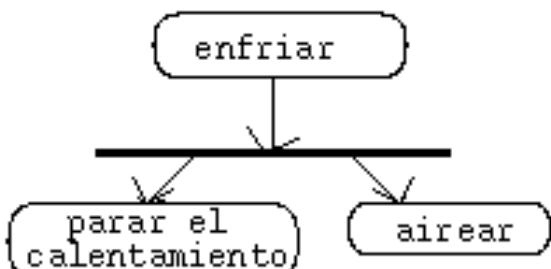
Diagramas de actividades

- ☞ Es una variante de los diagramas de estado-transición, organizados en relación a las acciones y destinado principalmente a la representación del comportamiento interno de un método (la realización de una operación) o de un caso de uso.
- ☞ Cada actividad representa una etapa particular en la ejecución del mecanismo.
- ☞ Las transiciones, representadas por flechas, son automáticas

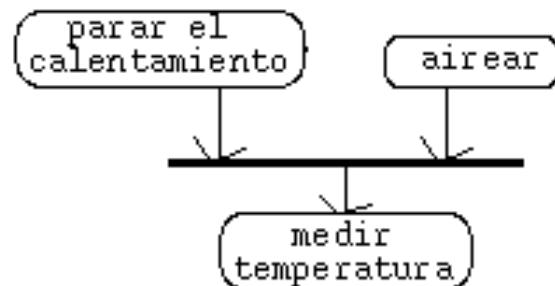
De actividades ...



Para representar condiciones y decisiones

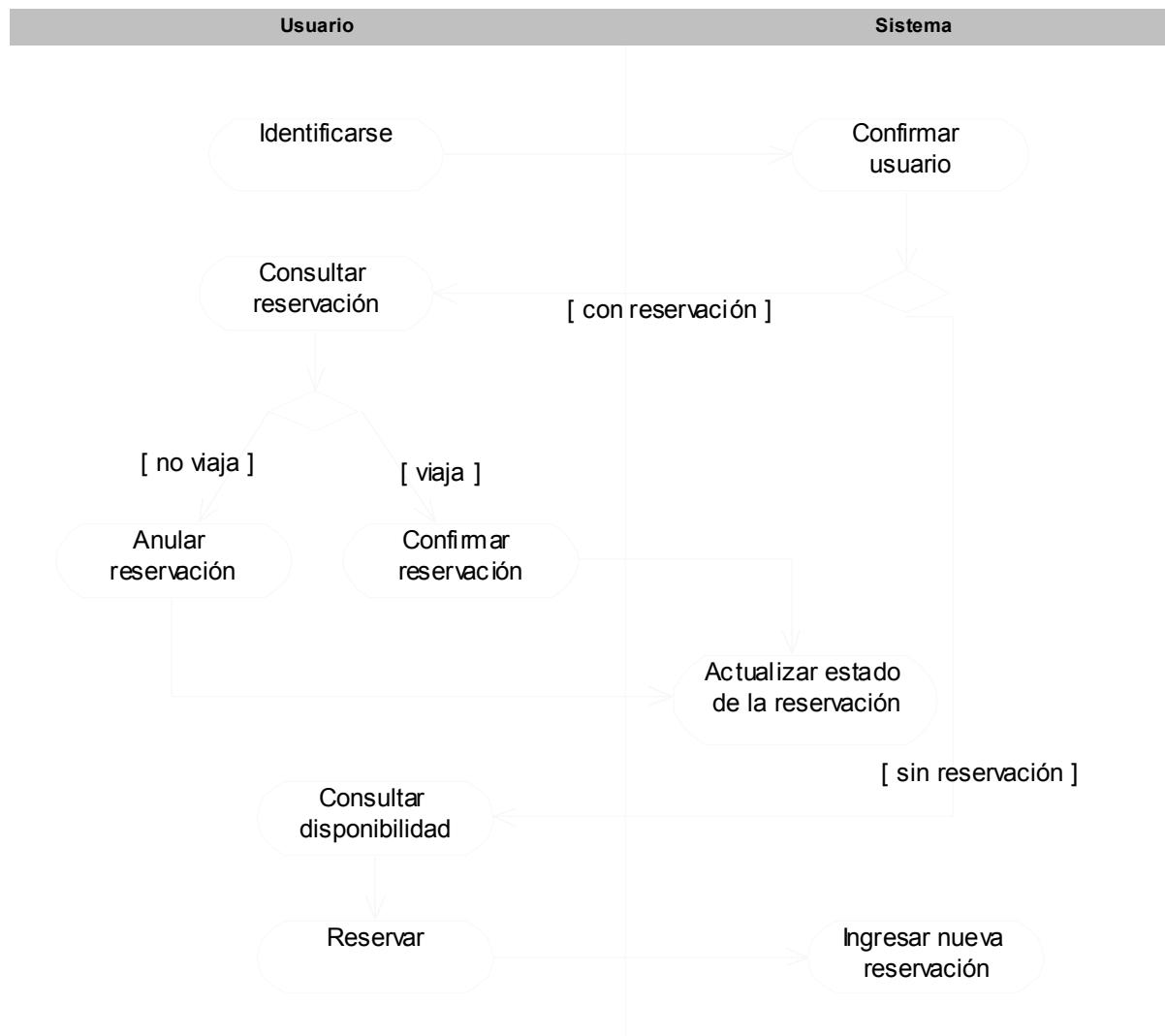


Sincronización de flujos paralelos



Fusión de flujos paralelos

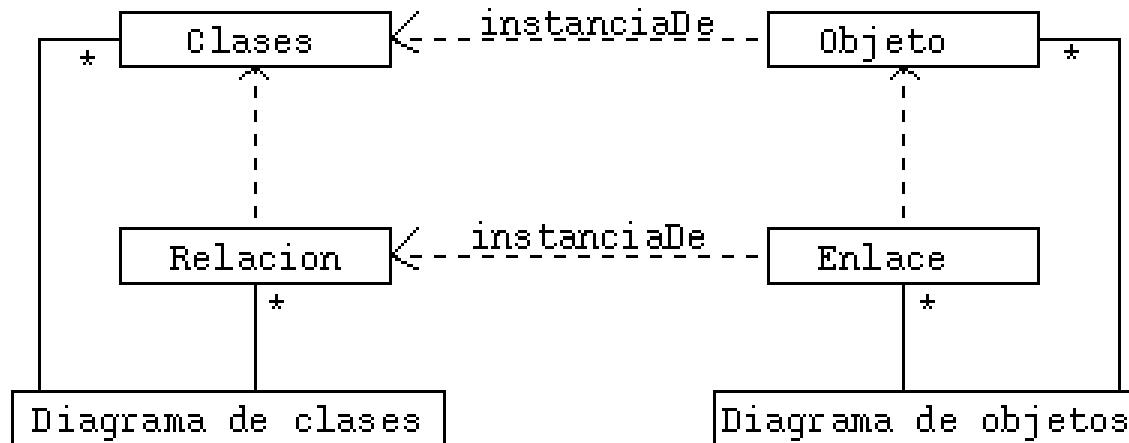
De actividades



Diagramas de objetos ...

- ☞ Los diagramas de objetos o de instancias muestran los objetos y sus enlaces.
- ☞ Ellos representan la estructura estática, como el diagrama de clases.
- ☞ La notación es la misma que para los diagramas de clases donde los elementos que son instancias se subrayan.
- ☞ Ellos se utilizan principalmente para mostrar un contexto o para facilitar la comprensión de estructuras complejas, como las recursivas

Metamodelo

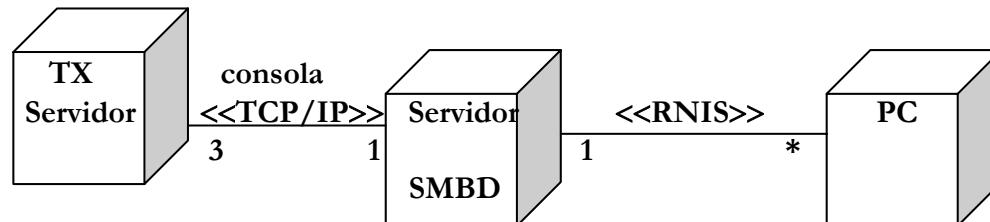


Diagramas de despliegue ...

- ☞ Muestran la disposición física del hardware (nodos) que conforman la composición de un sistema y la distribución de los programas ejecutables en dicho hardware.
- ☞ Todo sistema se describe por un pequeño número de diagramas de despliegue y normalmente, uno solo es suficiente.
- ☞ La naturaleza del equipo puede ser expresada por medio de un estereotipo, i.e. <<Dispositivo>>, <<Procesador>> o <<Memoria>>.

De despliegue ...

- ☞ Los cubos pueden relacionarse por asociaciones, a priori bidireccionales, donde la naturaleza del soporte se puede indicar con un estereotipo, i.e. <<Conexión>>, <<TCP/IP>>, etc.
- ☞ Un determinado cubo puede representar clases de hardware o uno en específico, colocando su nombre subrayado.



Diagramas de componentes ...

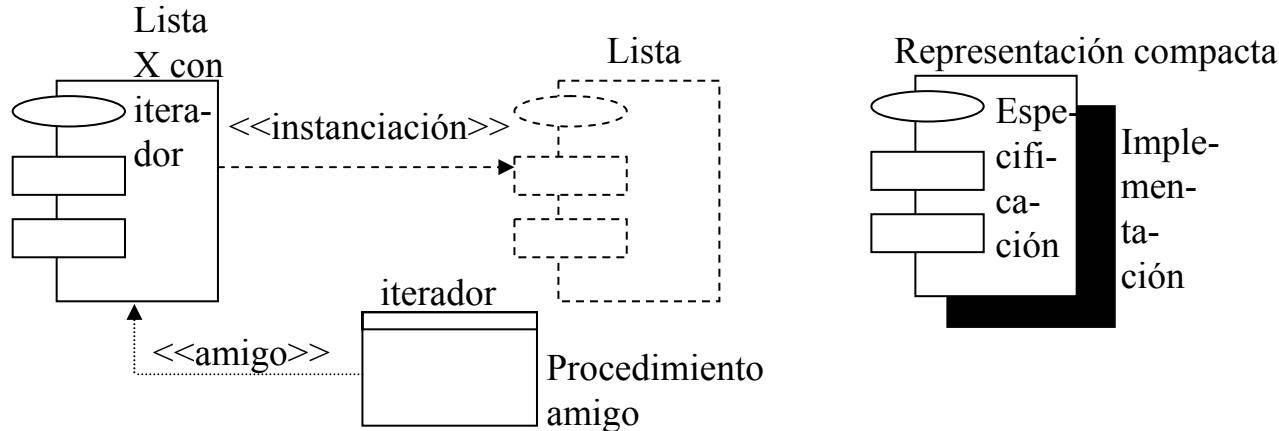
- ☞ Diagramas de módulos y procesos.
- ☞ Describen los elementos físicos y sus relaciones en el ambiente de implementación del sistema.
- ☞ Los módulos representan toda suerte de elementos físicos que se utilizan en el desarrollo de software.
Pueden ser: archivos simples, paquetes o bibliotecas.
- ☞ Por omisión, cada clase del modelo lógico se realiza con dos componentes: la especificación y la implementación o cuerpo.

De componentes ...



- ☞ La especificación contiene la interfaz de la clase y la implementación contiene la realización de la misma clase.
- ☞ La especificación puede ser genérica en el caso de las clases parametrizables. En C++, la especificación se corresponde con el archivo .h y la implementación con el archivo .cpp o .cc

De componentes ...



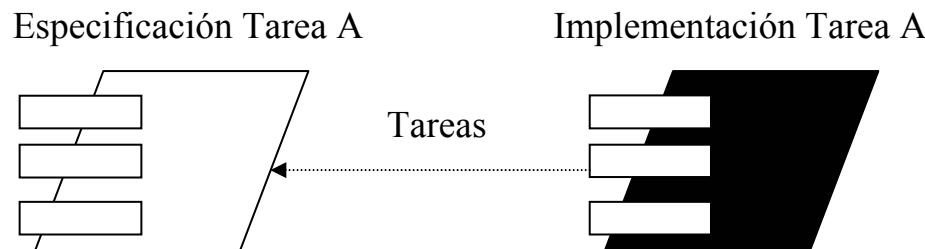
☞ Dependencias entre componentes:

- ☞ Indican cuando una componente hace uso de los servicios de otra.
- ☞ Se utiliza la relación de dependencia desde la componente que utiliza hasta la otra que ofrece el servicio.
- ☞ Por lo general, ella representa la dependencia de compilación entre las componentes, dando el orden de compilación según el grafo de las relaciones de dependencia.
- ☞ Esta relación puede tener un estereotipo para indicar la naturaleza de la implementación, por ejemplo: <<amiga>>.

De componentes ...

☞ Procesos y tareas:

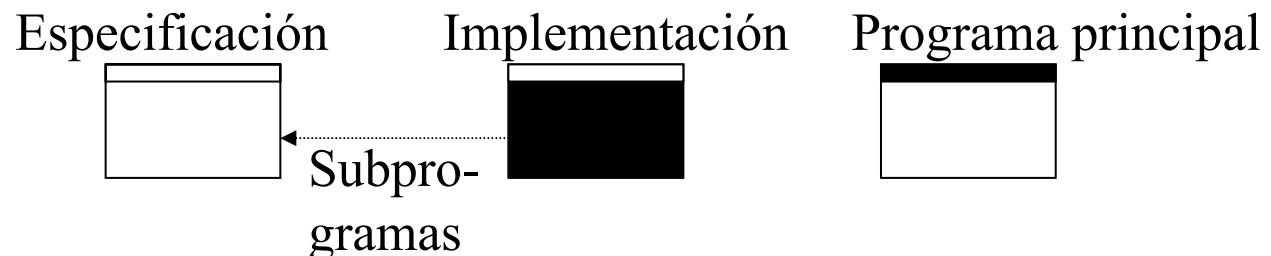
- ☞ Las tareas corresponden a aquellas componentes que tienen su propio flujo de control (thread) y pueden estar contenidas en otras componentes.
- ☞ Se predefinen como estereotipos: <<Proceso>> o <<Flujo>>. Varios flujos pueden compartir el mismo espacio de direccionamiento dentro de un proceso.



De componentes ...

☞ Programas:

- ☞ Los programas principales que son los puntos de entrada en las aplicaciones se identifican con un nombre que se utiliza para asociar el programa ejecutable con la aplicación.
- ☞ Esto permite asociar el modelo de componentes con el modelo de procesos.
- ☞ Los subprogramas reagrupan los procedimientos y las funciones que no pertenecen a ninguna clase.
- ☞ Las componentes pueden contener las declaraciones de los tipos básicos necesarios para la compilación de los subprogramas, pero no contienen clases.

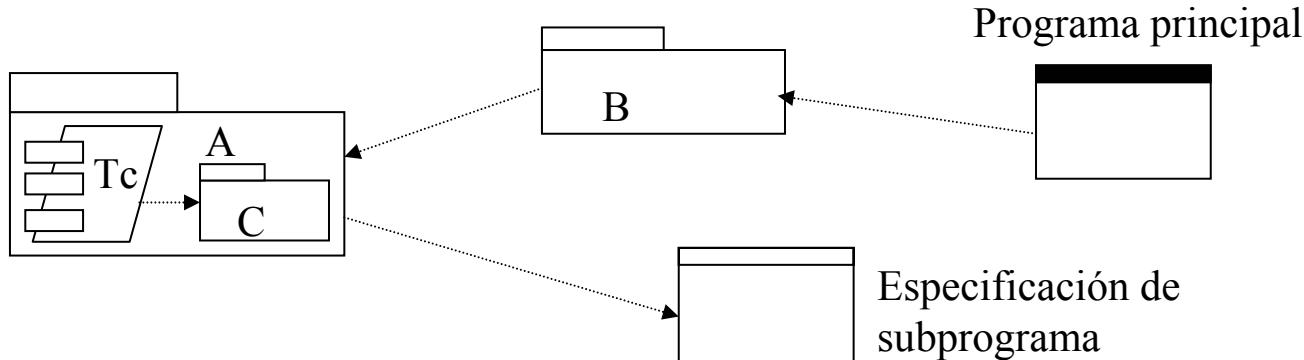


De componentes ...

- ☞ Subsistemas o componentes:
 - ☞ **La descomposición en subsistemas *no* es una descomposición funcional.**
 - ☞ **Las funciones del sistema se expresan desde el punto de vista del usuario descrita en los casos de uso, que se traducen en interacciones entre los objetos de las clases que están encapsuladas en las categorías.**
 - ☞ **Los objetos que realizan las interacciones se distribuyen en las diferentes categorías, el código correspondiente se almacena en los módulos y en los subsistemas.**

De componentes

- ☞ Los subsistemas poseen una parte pública y otra privada, salvo por indicación explícita, todo módulo dentro de un subsistema es visible.
- ☞ Una componente puede depender de un subsistema.



Diagramas de estado-transición ...

- ☞ Visualizan los autómatas de estado finito, desde el punto de vista de los estados y las transiciones.
- ☞ Un autómata es una abstracción de los comportamientos posibles.
- ☞ El comportamiento de los objetos de una clase pueden ser descritos de manera formal en términos de los estados y los eventos por medio de un autómata ligado a la clase.

De estado-transición ...

- ☞ Los objetos que no presentan un comportamiento reactivo muy marcado se pueden considerar como objetos que siempre están en un mismo estado.
- ☞ Los autómatas de UML son deterministas y se basan en autómatas jerárquicos que poseen los conceptos de ortogonalidad, agregación y generalización.
- ☞ Cada objeto sigue globalmente el comportamiento descrito en el autómata asociado a su clase y se encuentra en un momento dado en un estado que caracteriza sus condiciones dinámicas

De estado-transición ...

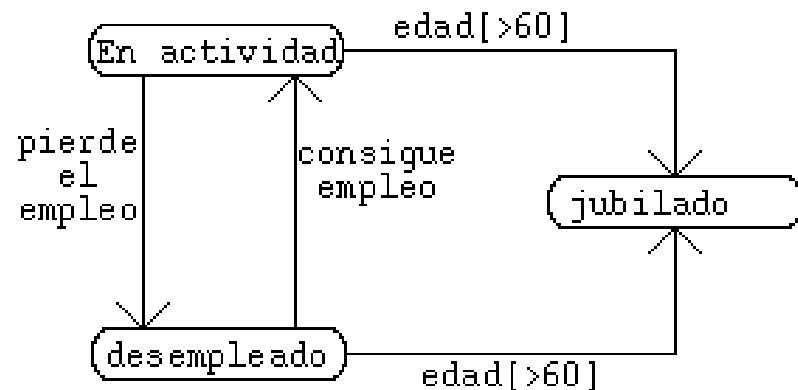
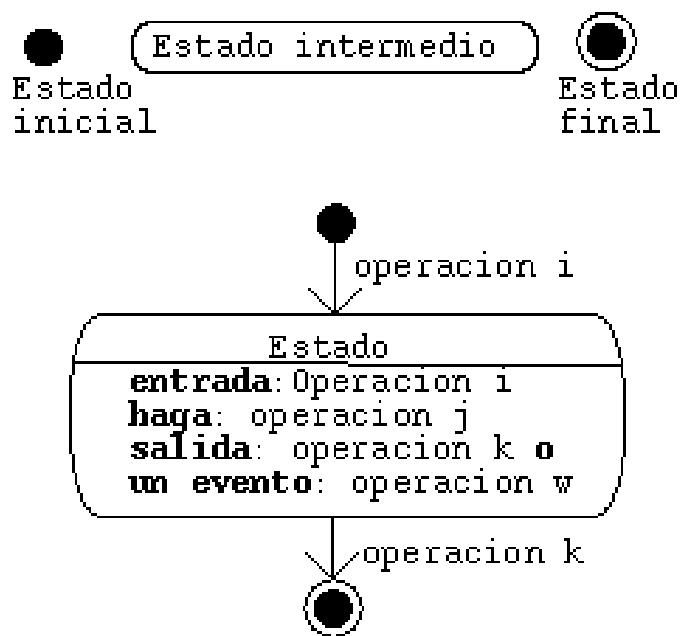
☞ Estados y transiciones:

- ☞ Un estado es siempre la imagen de la conjunción instantánea de los valores contenidos en sus atributos y la presencia o no de sus enlaces a otro objetos.
- ☞ Cada objeto está en un momento dado, en un estado particular.
- ☞ Un objeto está siempre en un estado dado por un cierto tiempo y no puede estar en un estado desconocido o indefinido.
- ☞ Un diagrama de estado-transición no puede ser ambigüo, siempre debe tener un estado inicial y puede tener solo un estado final, que puede corresponder a condiciones de finalización diferentes.

De estado-transición ...

- ☞ En aquellos sistemas que no se paran jamás, no existirá ningún estado.
- ☞ Los diagramas de estado-transición son digrafos, donde los estados son los nodos ligados entre sí por líneas bidireccionales, denominadas transiciones.
- ☞ El paso de un estado a otro se efectúa luego que una transición se dispara por un evento que ocurre en el dominio del problema.
- ☞ El paso de un estado a otro es instantáneo, pues el sistema tiene que estar siempre en un estado conocido.
- ☞ Las transiciones pueden ser reflexivas (ligadas al mismo estado)

De estado-transición ...



De estado-transición ...

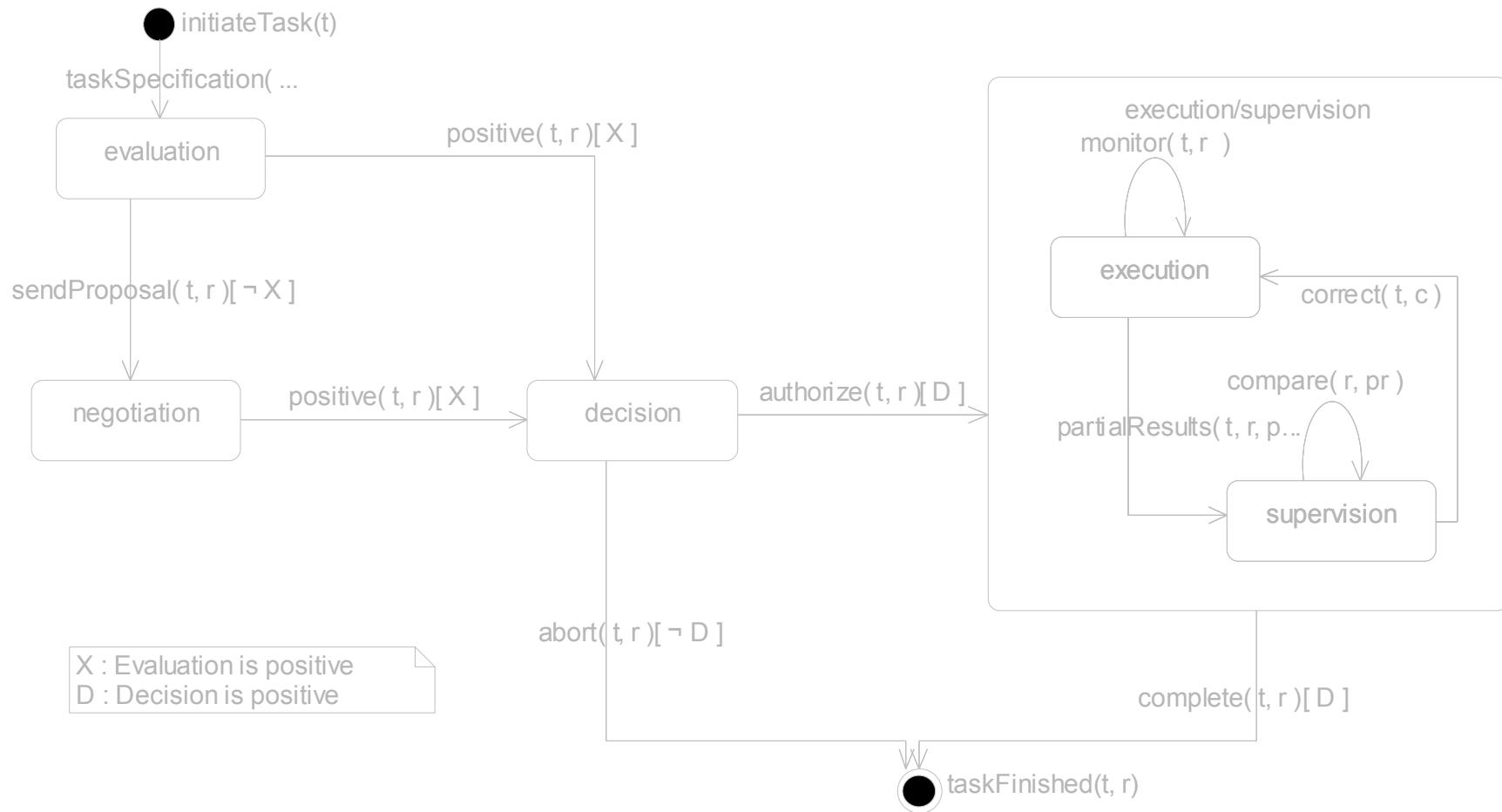
☞ Eventos:

- ☞ Un evento se corresponde con la ocurrencia de una situación dada en el dominio del problema, es por naturaleza una información instantánea que debe ser tratada en forma inmediata, sin espera.
- ☞ Un evento sirve de disparador para pasar de un estado a otro, donde las transiciones indican el camino en el digrafo de estados y los eventos determinan que camino seguir.
- ☞ Los eventos, las transiciones y los estados son elementos indisociables en la descripción del comportamiento dinámico.

De estado-transición ...

- ☞ Los objetos se comportan como elementos pasivos controlados por los eventos que provienen del sistema.
- ☞ Sintaxis de un evento:
 $\text{NombreDelEvento}(\text{nombreDelParámetro} : \text{tipo}, \dots)$
- ☞ La especificación completa de un evento comprende:
 - El nombre del evento*
 - La lista de parámetros*
 - El objeto fuente*
 - El objeto destino*
 - La descripción del significado del evento.*

De estado-transición ...



De estado-transición ...

- ☞ Puntos de ejecución para especificar las operaciones:
 - ☞ *la acción asociada a una transición de entrada*
 - ☞ *la acción de entrada al estado*
 - ☞ *la actividad dentro del estado*
 - ☞ *la acción de salida del estado*
 - ☞ *la acción asociada a los eventos internos*
 - ☞ *la acción asociada a la transición de salida del estado*
- ☞ Por omisión, un autómata no tiene memoria, la notación H en la esquina inferior izquierda del diagrama del estado, ofrece un mecanismo para memorizar el último subestado visitado y para volver a él luego de una nueva entrada al estado.
- ☞ Una memorización en subestados anidados se indica con H*

De estado-transición ...

- ☞ Comunicación entre objetos y transiciones temporizadas:
 - ☞ Es posible el envío de mensajes producidos por un evento a cualquier conjunto de objetos. Las formas más comunes son el envío a todos los objetos (difusión) o a un objeto particular (punto a punto).
 - ☞ En el caso de un autómata que describe una clase compuesta, las acciones pueden hacer referencia directa a las operaciones declaradas en la clase.

De estado-transición

- ☞ Las esperas son por definición actividades que duran un cierto tiempo.
- ☞ Una espera se asocia a un estado en vez de a una transición y se representa por medio de una actividad de espera, la cual se interrumpe luego que el evento esperado se dé.
- ☞ Las temporizaciones pueden representarse directamente sobre la transición disparada después del retardo de espera, según: tempo(duración).

Metamodelo

