# Flujo de trabajo: Diseño

# Contenidos

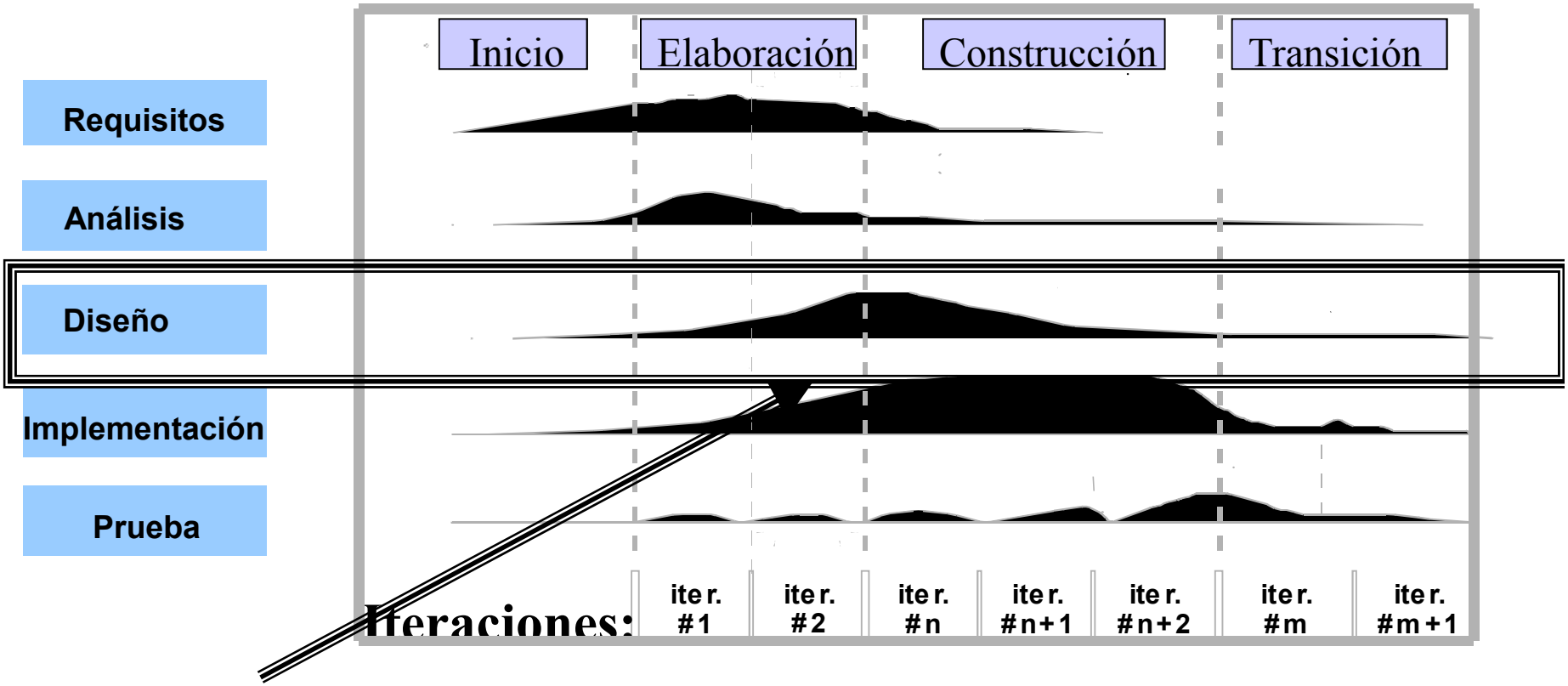
- 1.- Introducción
- 2.- El rol del diseño dentro del CV
- 3.- Artefactos
- 4.- Dos posibles patrones de diseño para acceder al nivel de datos (almacenamiento)
  - Usando un sistema OO
  - Usando un SGBD relacional

Anexos: modelo de despliegue, trabajadores y flujo de actividades

# 1. Introducción

- **Encontrar la forma (o solución) del sistema que cumpla con <u>todos</u> los requisitos (+ no funcion.)**
  - Modelo del análisis: comprensión de todos los requisitos.
- 1) Escoger herramientas (LP, SO, SGBD, GUI, concurrencia, distribución, componentes,…)
- 2) Obtener buena entrada a la fase de implementación
  - Que implementar sea directo a partir del diseño
- 3) Permitir implementación por varios equipos
  - Capturar las interfaces
  - Usar una notación común

# 2. Rol del Flujo de Trabajo de diseño en el CV

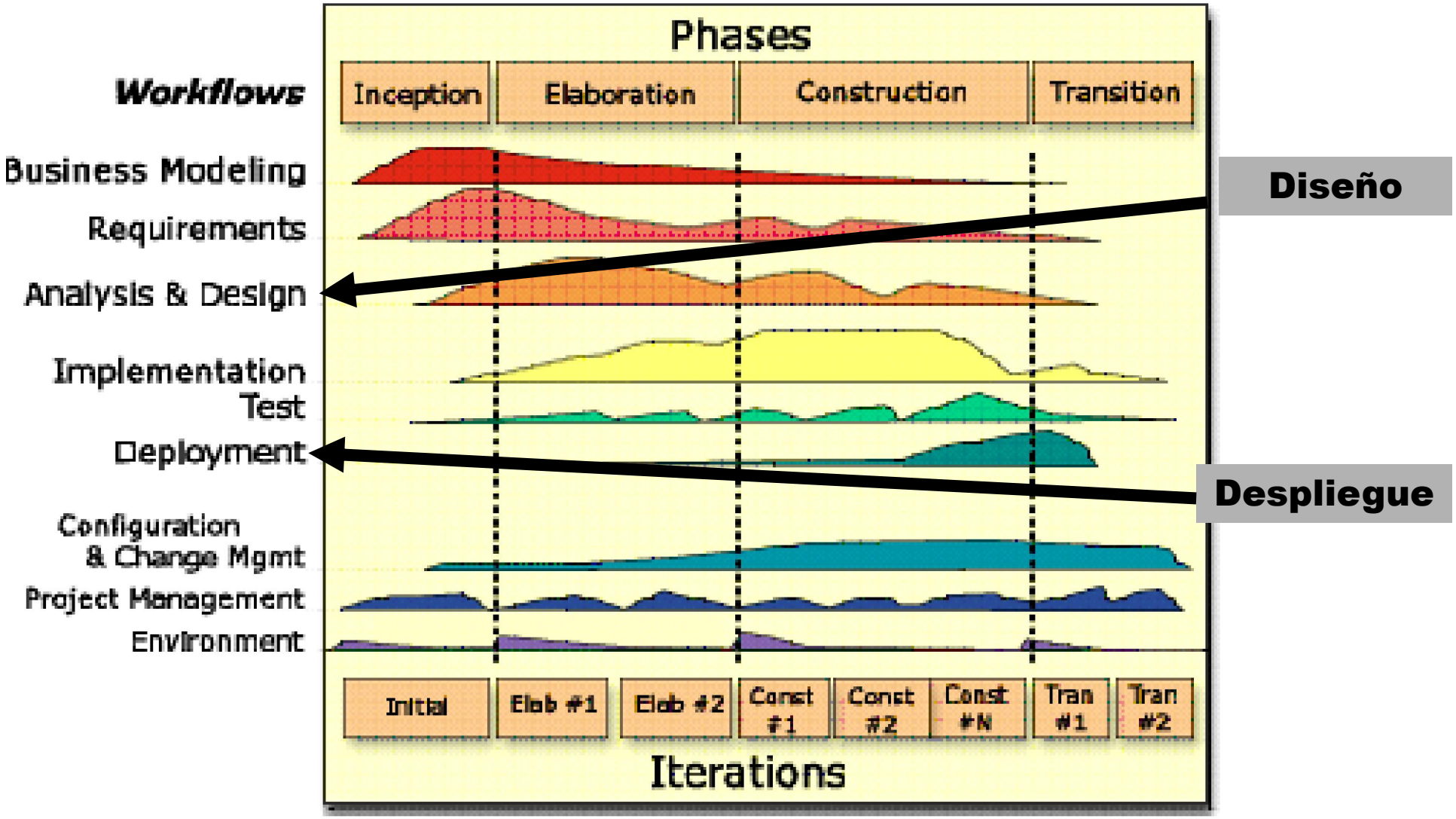



**Foco durante final de la fase de elaboración y comienzo de construcción**

- **obtener arquitectura estable**
- **y "anteproyecto" de implementación**

**El modelo de diseño SÍ se MANTIENE en todo el proyecto**

# El CV del PUD de Rational separa el “diseño” del “despliegue”



# 3. Artefactos a obtener en el FT de diseño

- Modelo del diseño
- Subsistema de diseño
- Clase de diseño
- Realización de caso de uso -- Diseño
- Interfaz
- Descripción de la arquitectura (vista del diseño)

**diseño**

- Modelo de despliegue
- Desc. de la arquitectura (vista del despliegue)

**despliegue**

**(NO LO TRATAREMOS)**

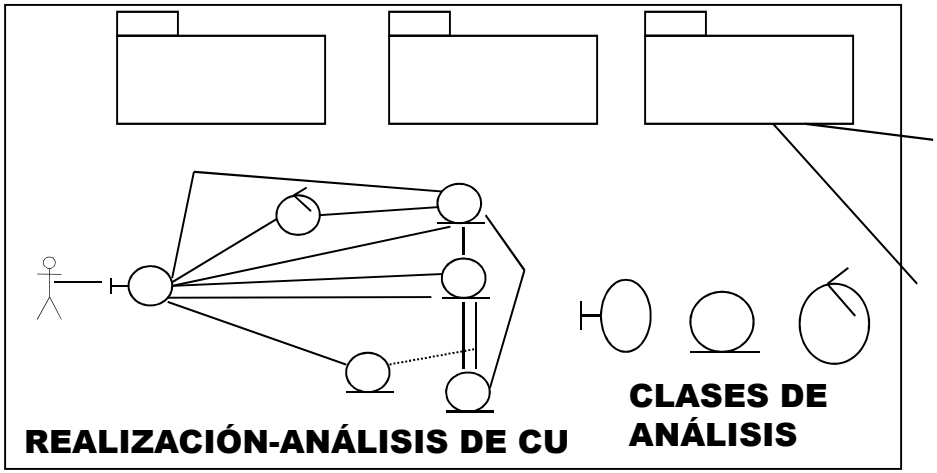
# JERARQUÍA DE SUBSISTEMAS DE DISEÑO

## MODELO DEL ANÁLISIS (MA) ⇒ MODELO DEL DISEÑO (MDiseño)

**PAQUETE DE ANÁLISIS**

**contiene**

- **Realizaciones de CU**
- **Clases de análisis**
- **Otros paquetes de análisis**



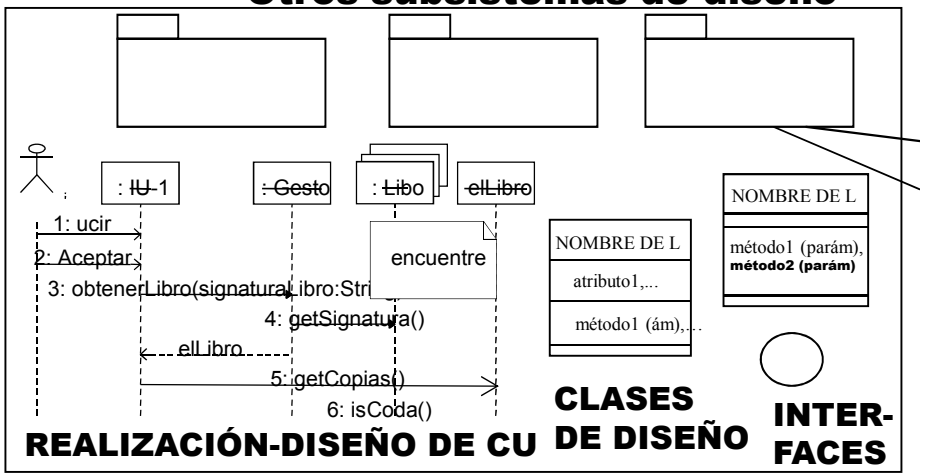
**REALIZACIÓN-ANÁLISIS DE CU**

**CLASES DE ANÁLISIS**

**SISTEMA DE DISEÑO**

**contiene**

- **Realizaciones de CU**
- **Clases de diseño**
- **Interfaces**
- **Otros subsistemas de diseño**




**REALIZACIÓN-DISEÑO DE CU**

**CLASES DE DISEÑO**

**INTER-FACES**

## + MODELO de DESPLIEGUE

# Artefacto: Clase de diseño

- Una clase de diseño es una abstracción de una clase (o constructor similar existente) en el LP
  - Operaciones, parámetros, atributos y tipos
  - Visibilidad de atributos y operaciones
  - Asociaciones y agregaciones (aunque luego se implementen añadiendo atributos)
  - Generalizaciones (con la semántica del LP utilizado)

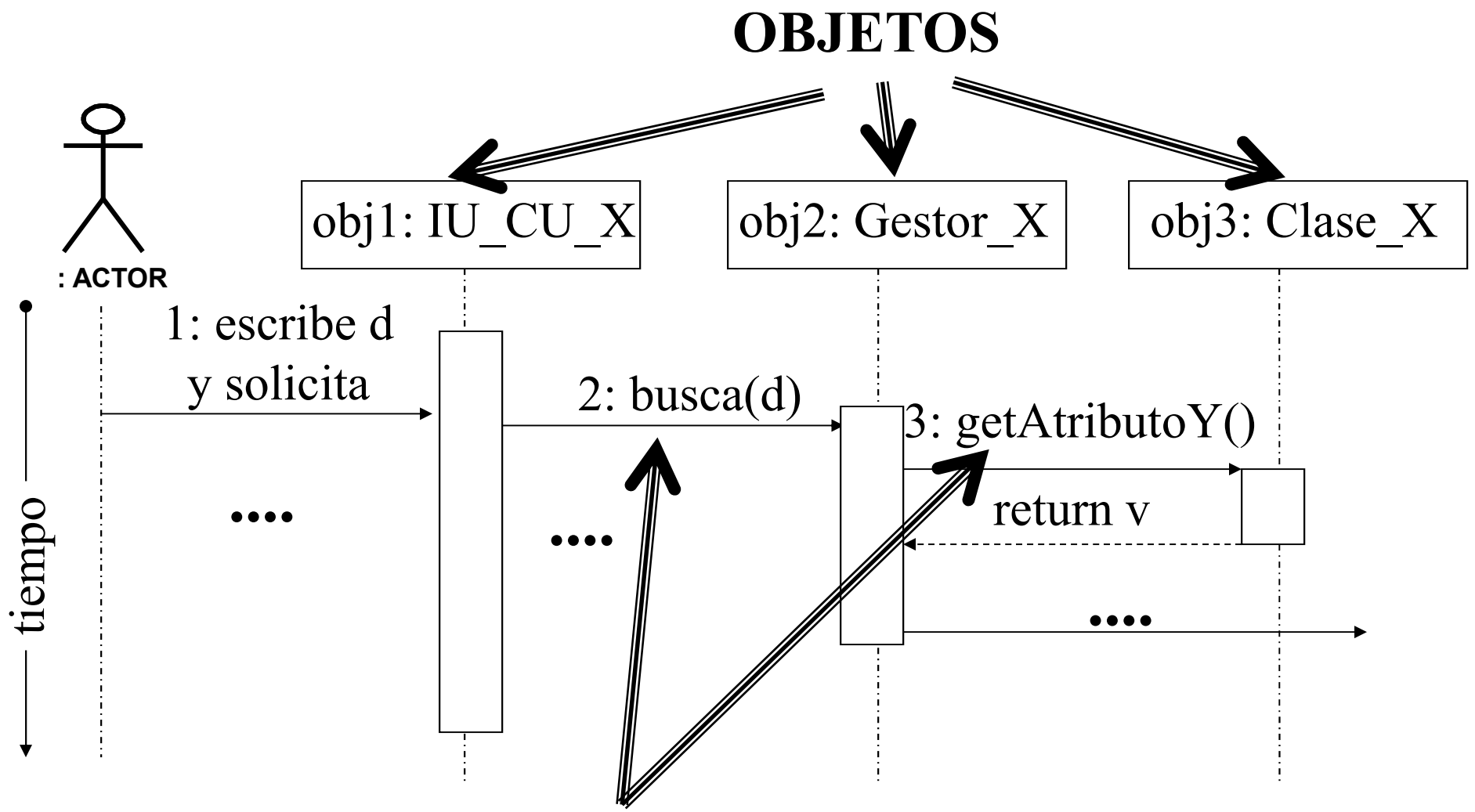
# Artefacto: Clase de diseño II

- Los métodos se especifican en lenguaje natural o en pseudocódigo
- Pueden especificarse requisitos de implementación de una clase
- Pueden ponerse estereotipos
  - "class module", "form", "user control" en VB
  - "interface" en Java

# Artefacto: Realización de CU -- diseño

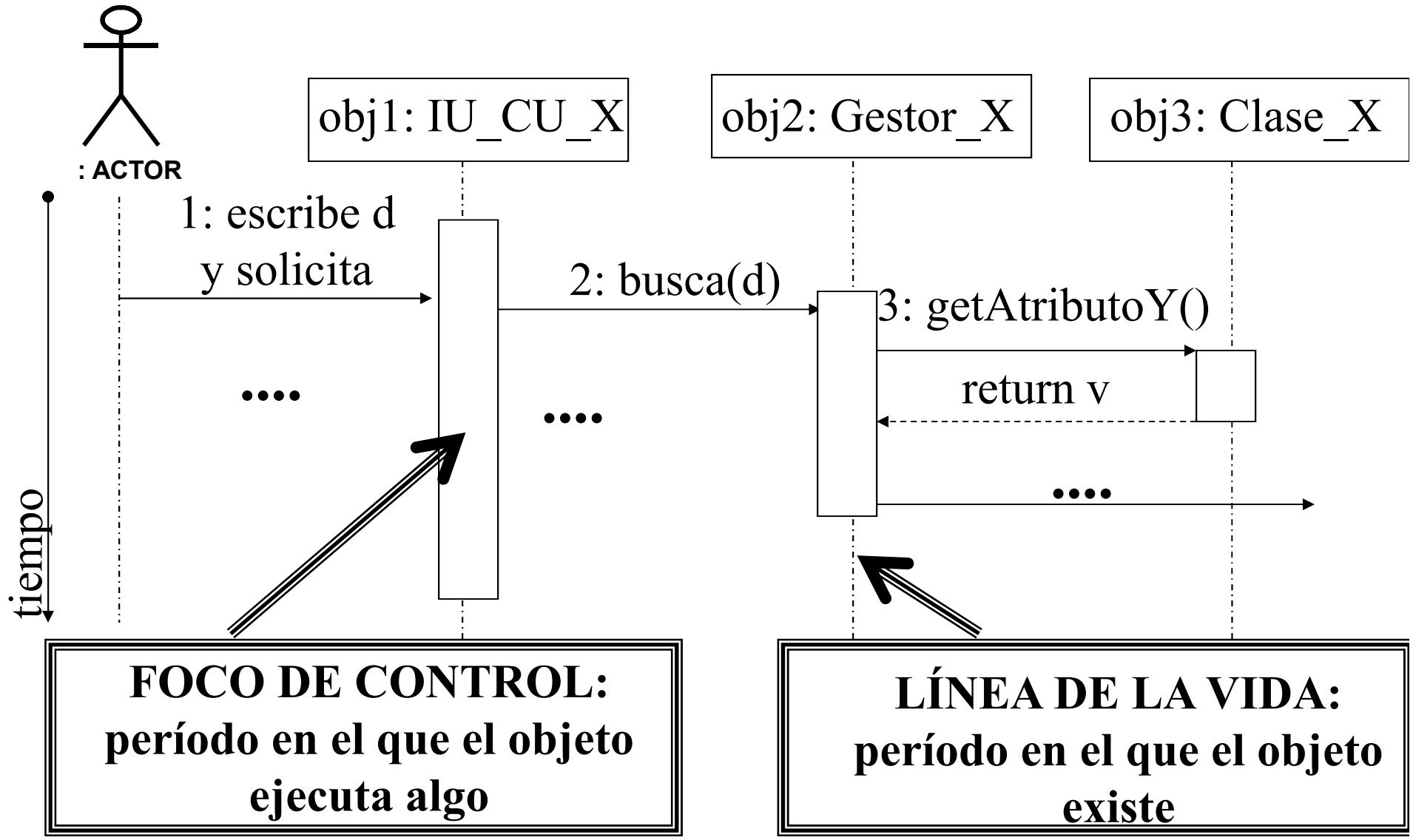
- Es una colaboración que indica cómo se realiza/ejecuta un CU, en términos de las clases de diseño y sus objetos
- Para cada CU habrá que añadir
  - El diagrama de secuencia
  - Flujo de eventos (en el diseño)
  - Requisitos de implementación

# Diagrama de Secuencia en UML




**PASO DE MENSAJES / LLAMADAS A MÉTODOS**

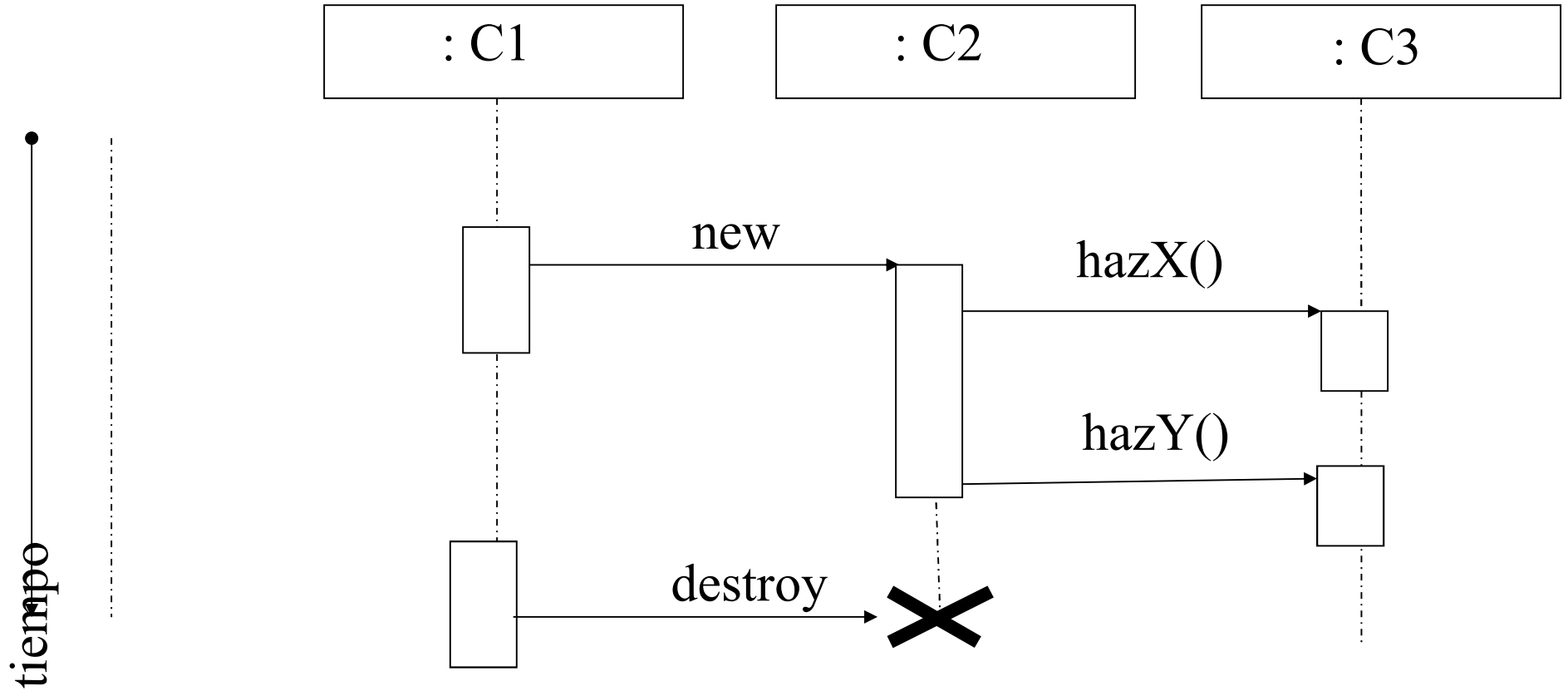
# Diagrama de Secuencia en UML





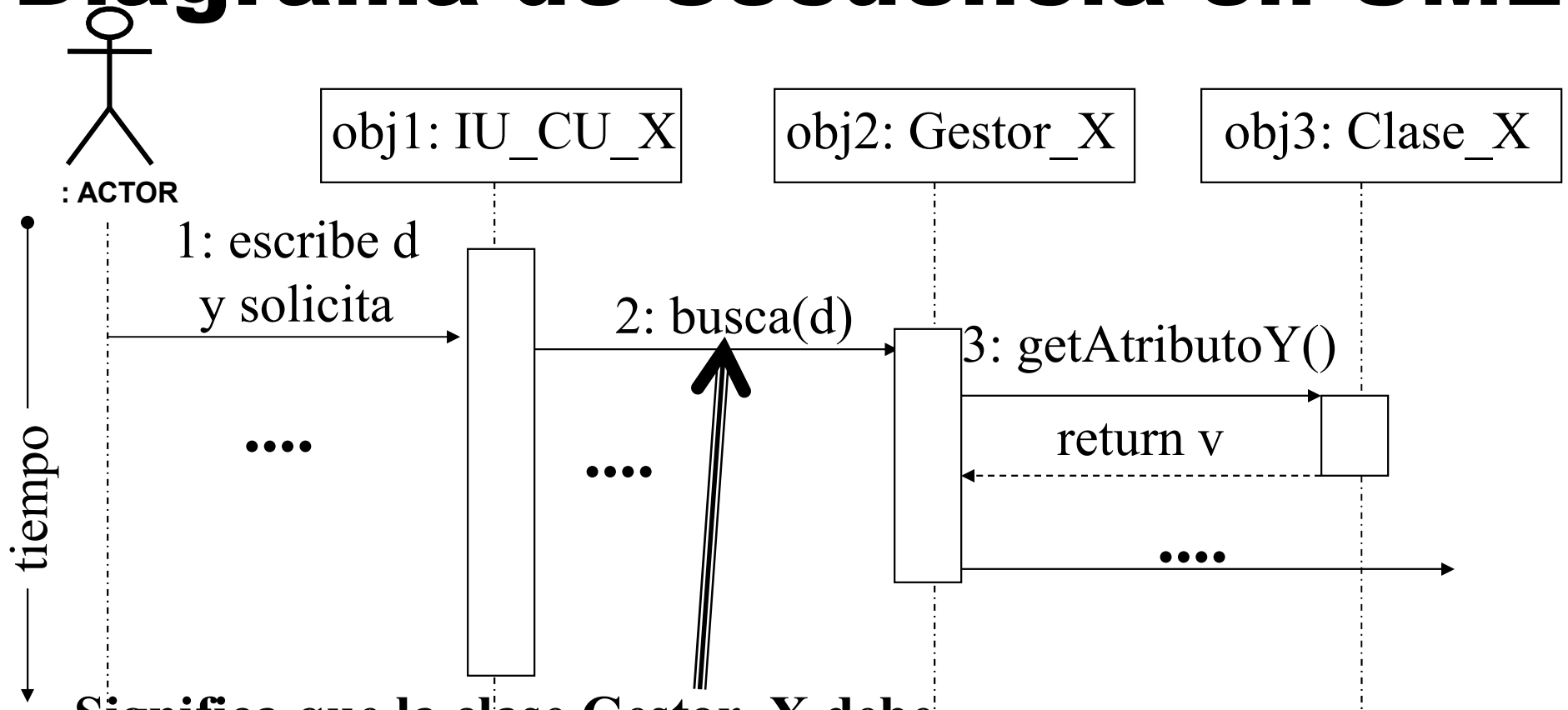

**Nota: no seremos muy rigurosos señalando el foco de control**

# Diagrama de Secuencia en UML

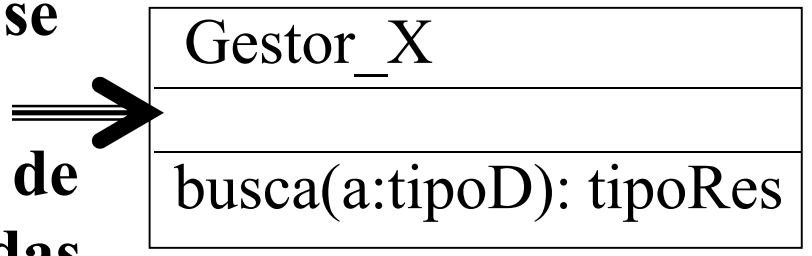



**Los objetos se pueden crear con el método NEW (comienza su línea de vida y foco de control mientras ejecutan cosas) y se pueden destruir con el método DESTROY (se termina su línea de vida)**

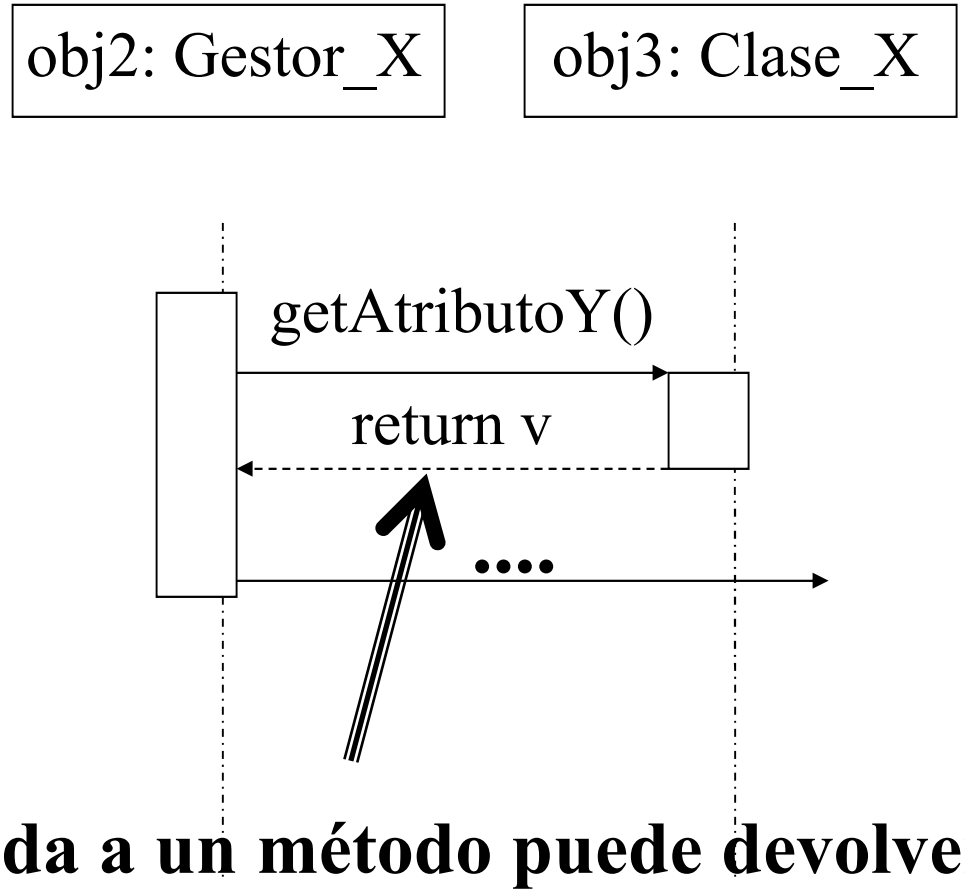
# Diagrama de Secuencia en UML




**Significa que la clase Gestor_X debe proporcionar el método busca. Así se DISEÑAN las clases, esto es, se identifican sus MÉTODOS a partir de las RESPONSABILIDADES definidas durante el FT del análisis**



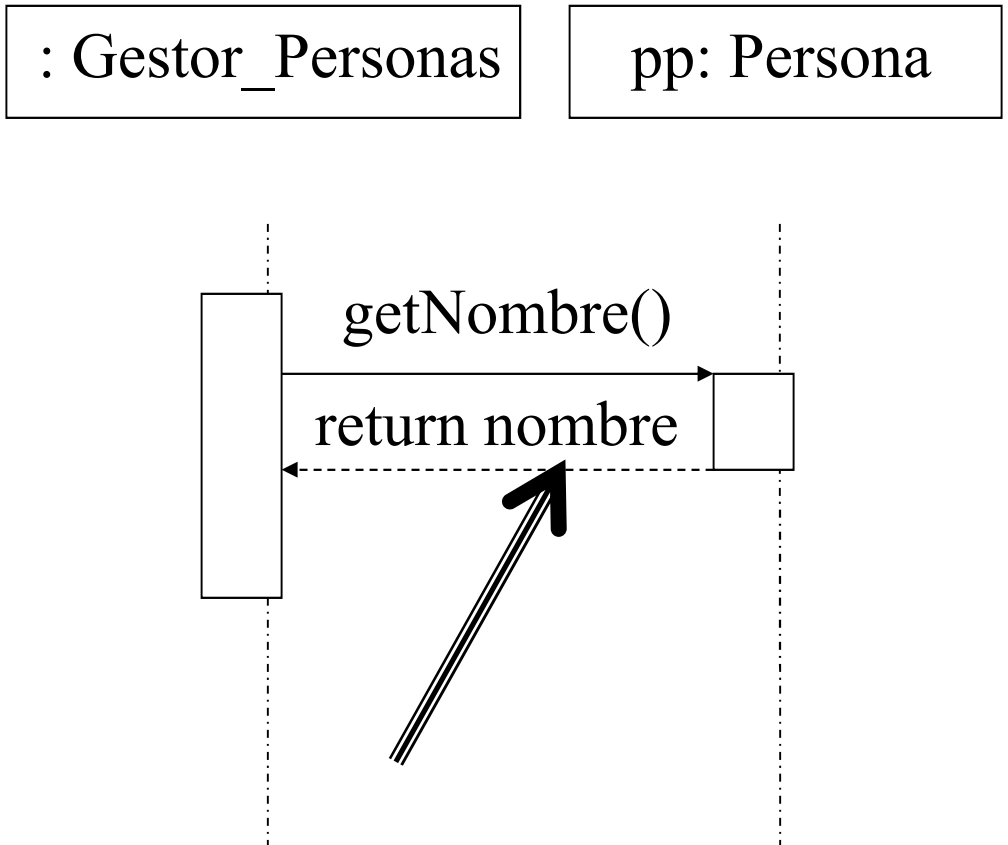
# Diagrama de Secuencia en UML




**Una llamada a un método puede devolver un valor (mensaje "return valor" en línea con puntos suspensivos).**

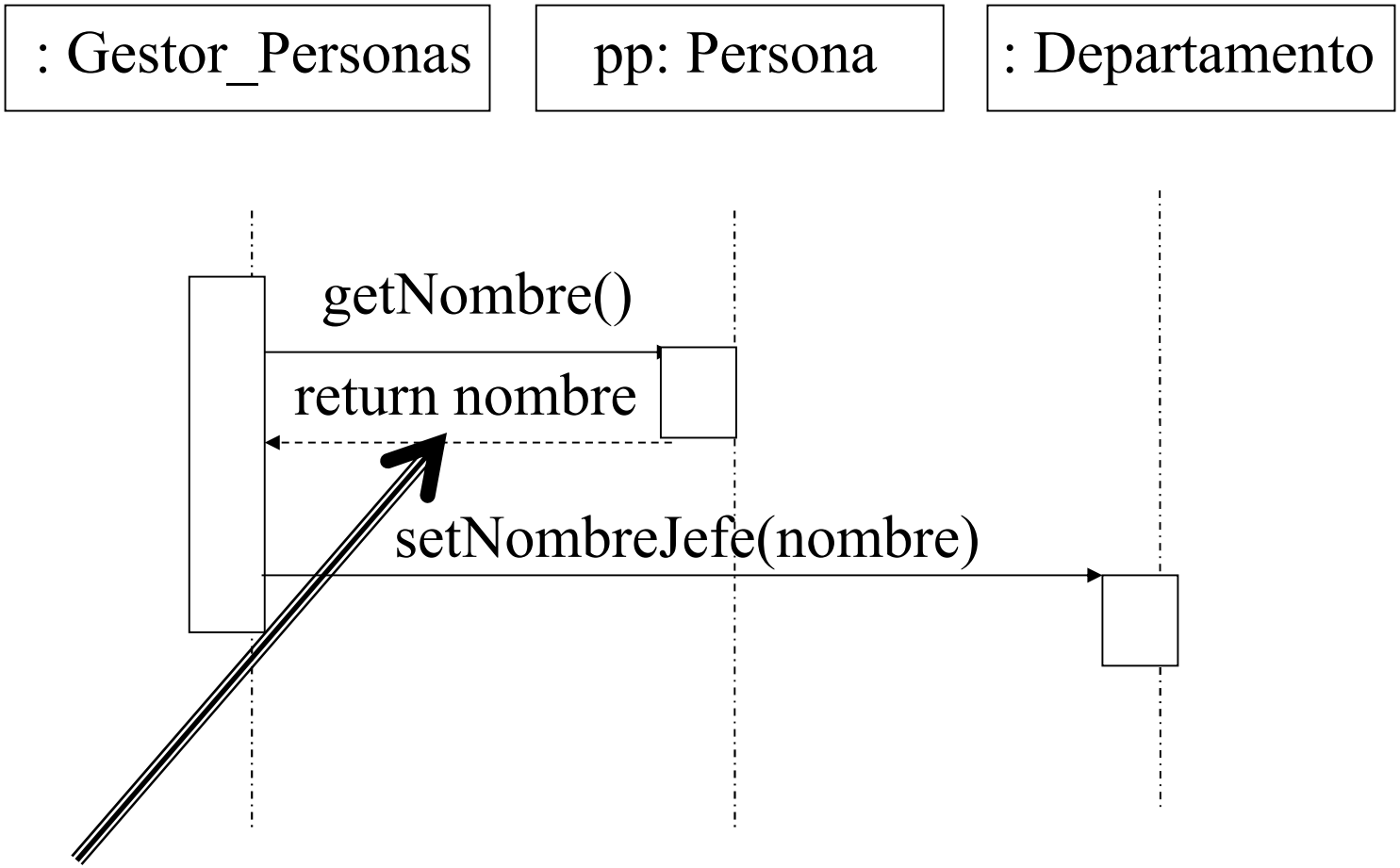
**Generalmente sólo se pondrá en el diagrama de secuencia cuando proporcione información interesante**

# Diagrama de Secuencia en UML



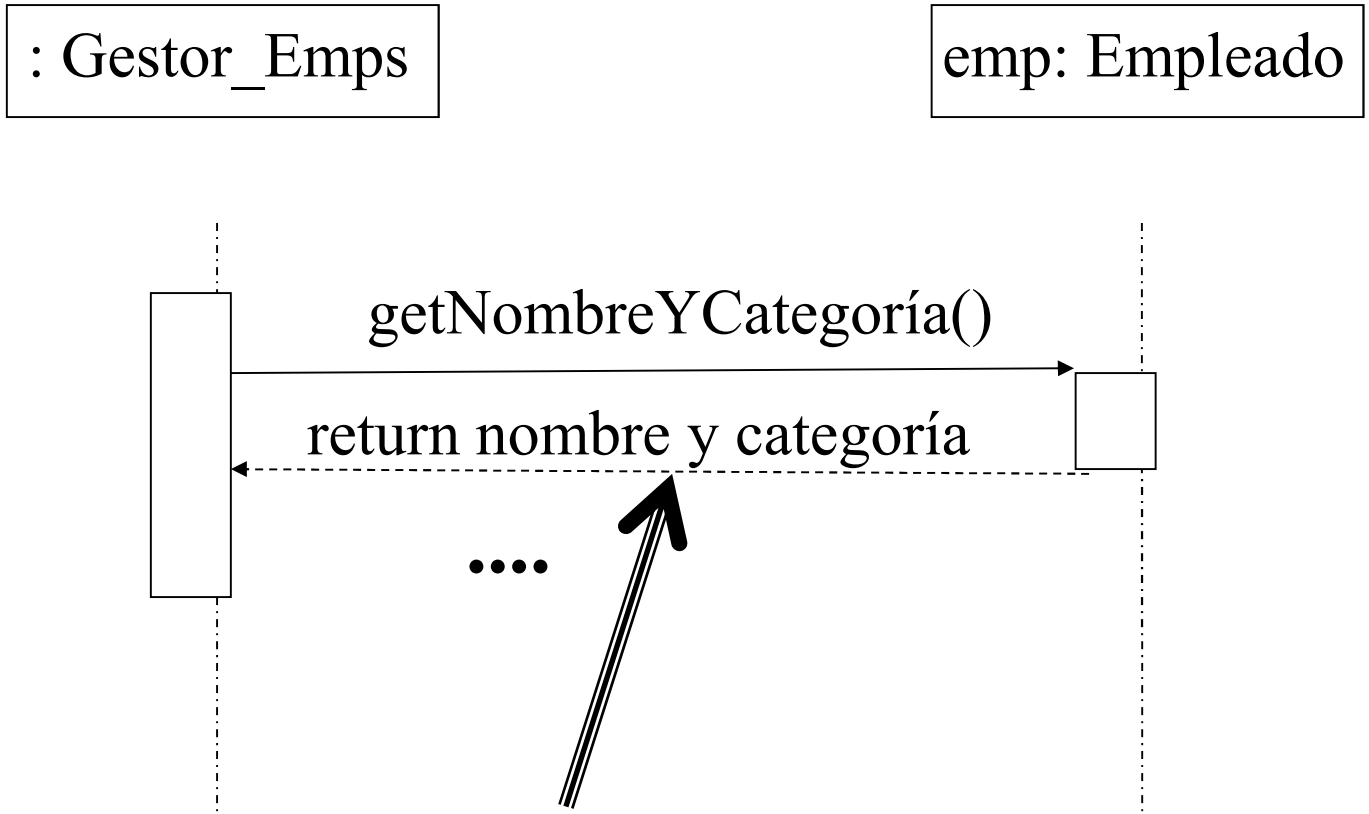

**En este caso NO ES NECESARIO. Es evidente que le estamos preguntando al objeto "pp" por su nombre, y eso será lo que nos devolverá. Nos ahorraremos una línea en el diagrama de secuencia y será más fácil de leer.**

# Diagrama de Secuencia en UML



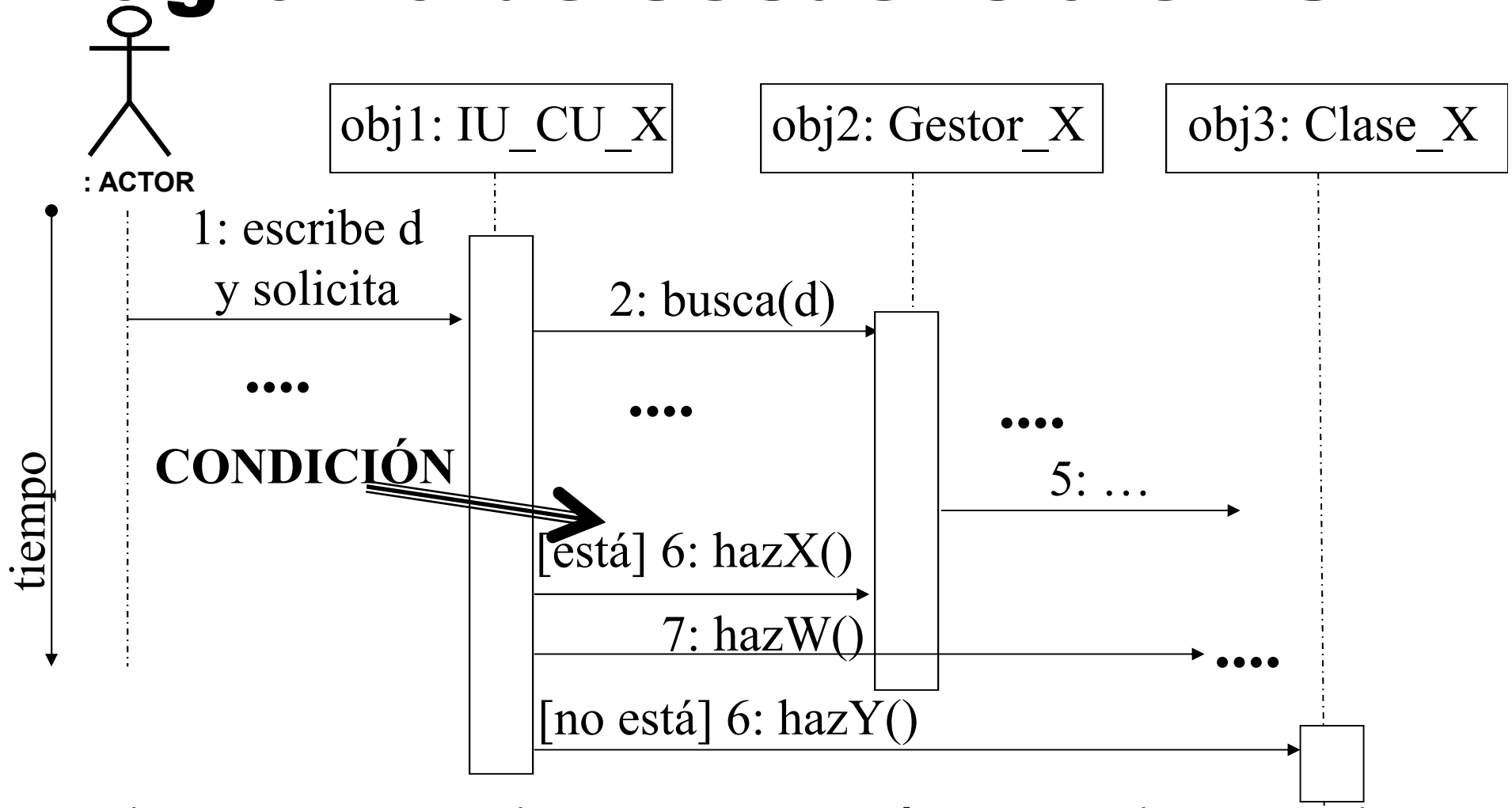

**En este caso SÍ ES NECESARIO. Así podemos ver que ponemos como nombre del jefe de departamento el nombre del objeto “pp”.**

# Diagrama de Secuencia en UML



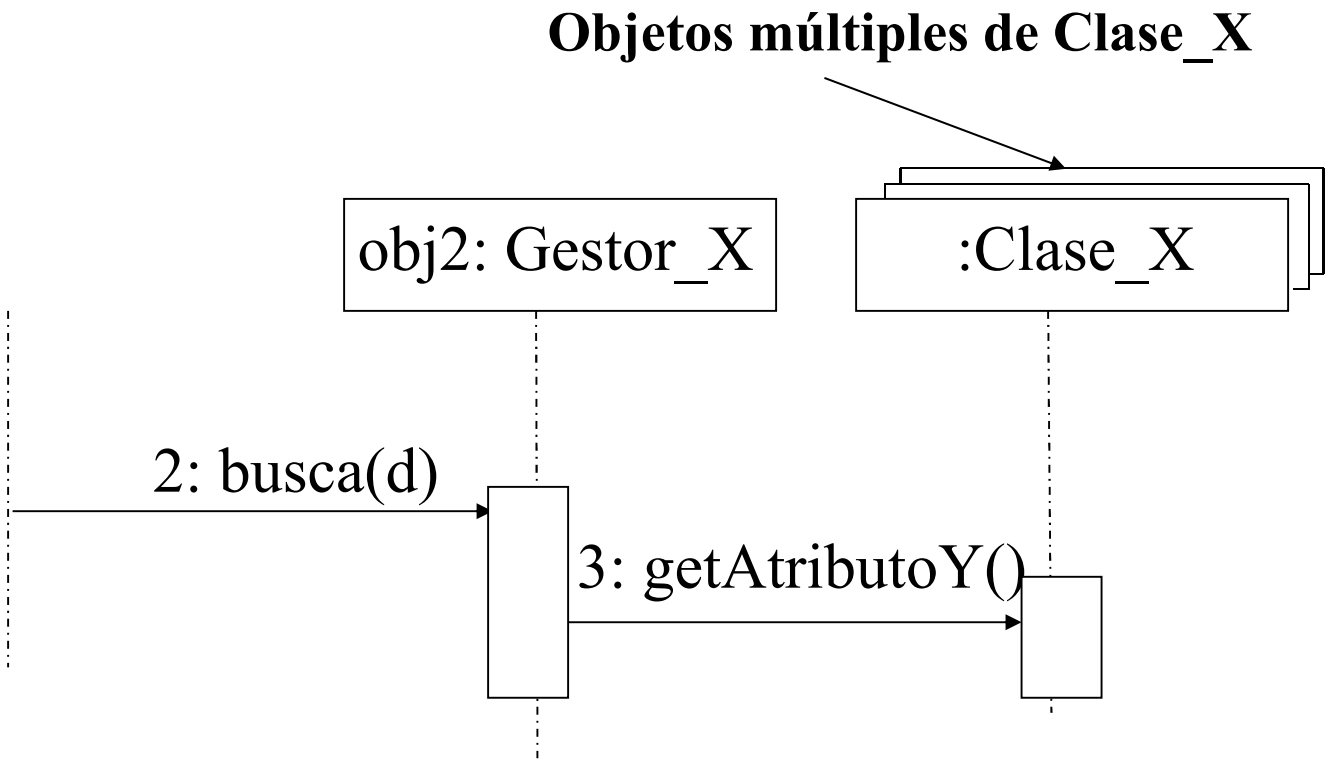

**Una llamada a un método NO puede devolver MÁS de un valor. Eso no es posible utilizando un lenguaje OO**

# Diagrama de Secuencia en UML



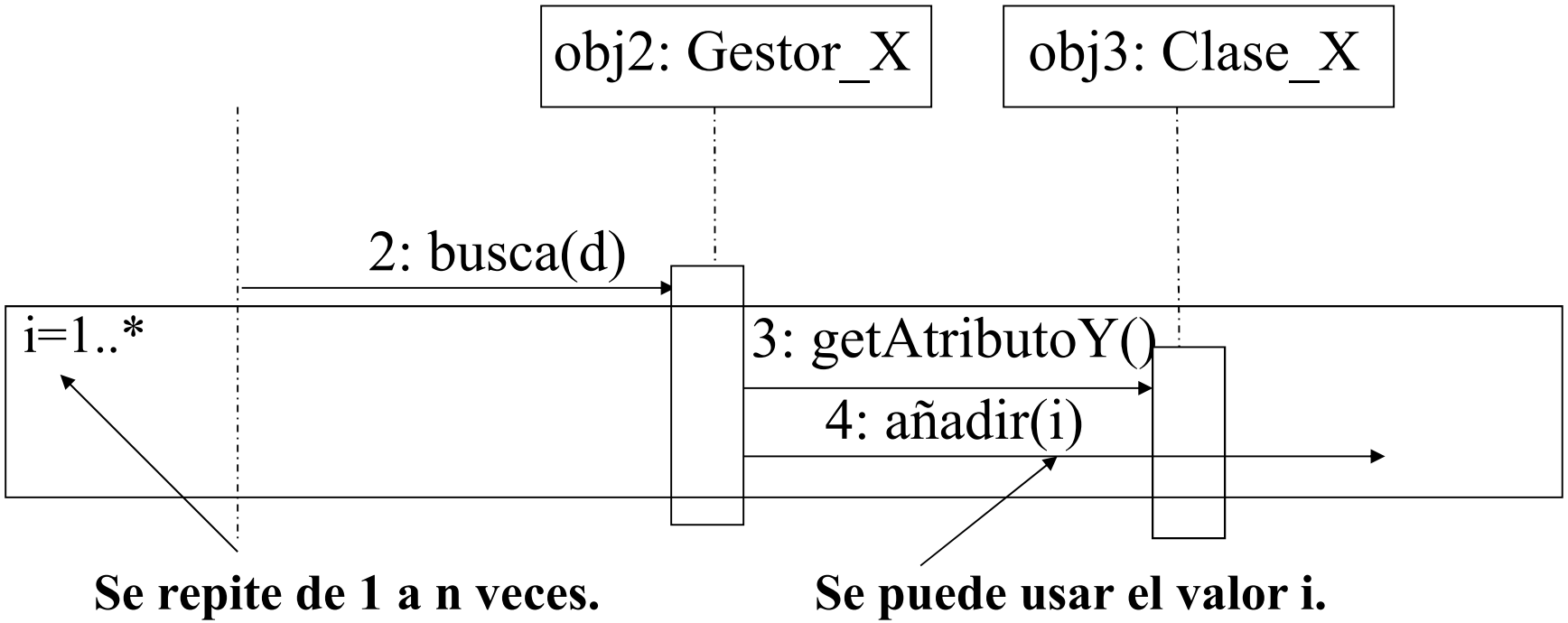

**Los diagramas de secuencias muestran los envíos de mensajes entre objetos <u>en orden secuencial</u> (se añaden números de secuencia 1: 2:, …). SE PUEDEN AÑADIR CONDICIONES. Los números de secuencia continúan independientemente en cada rama. NO HAY QUE ABUSAR DE LAS CONDICIONES. Es mejor realizar distintos diagramas de secuencia.**

# Diagrama de Secuencia en UML



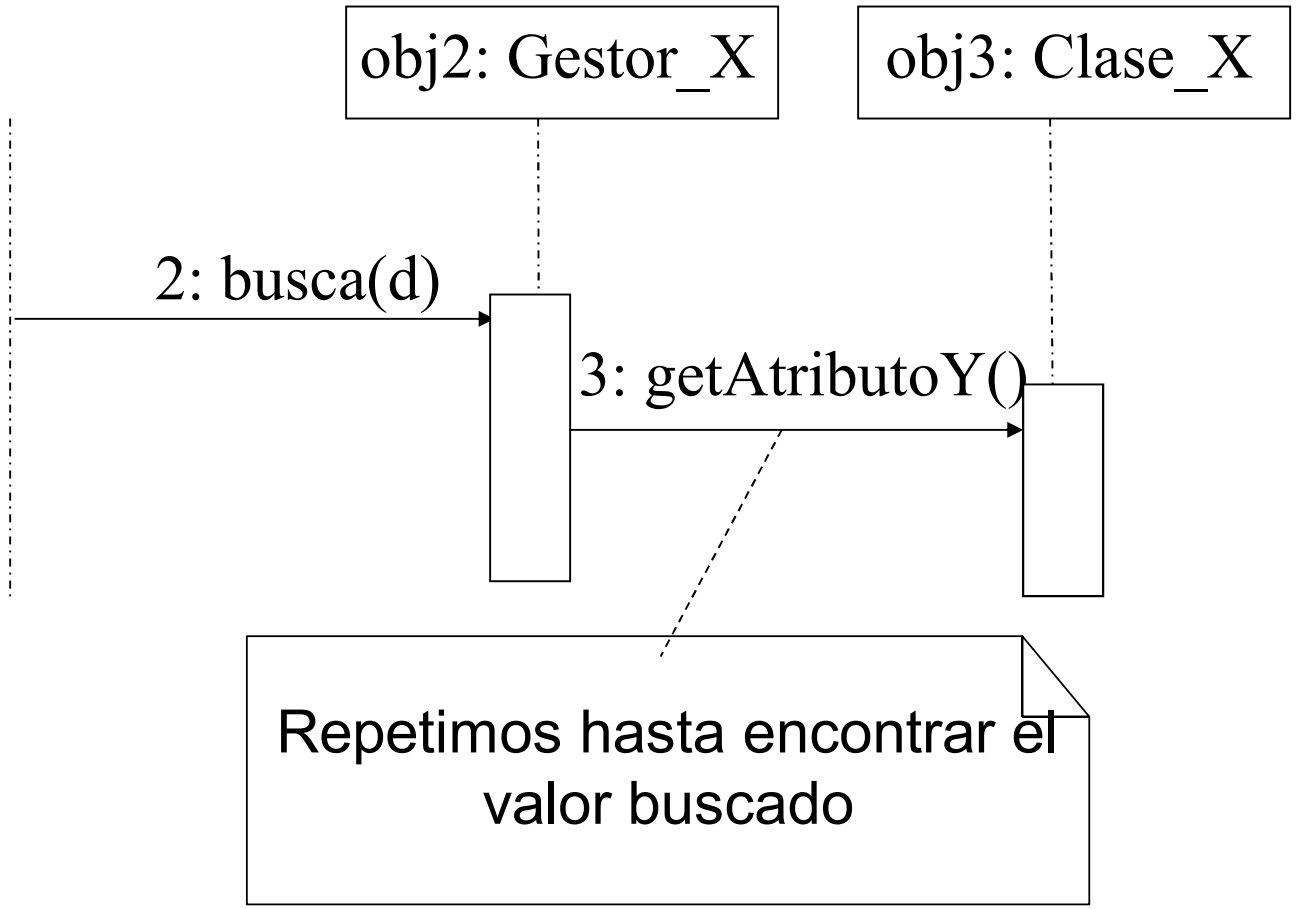

**SE PUEDE AÑADIR REPETICIONES. De esta manera se indica que a varios objetos de la clase Clase_X se les pide ejecutar el método getAtributoY()**

# Diagrama de Secuencia en UML



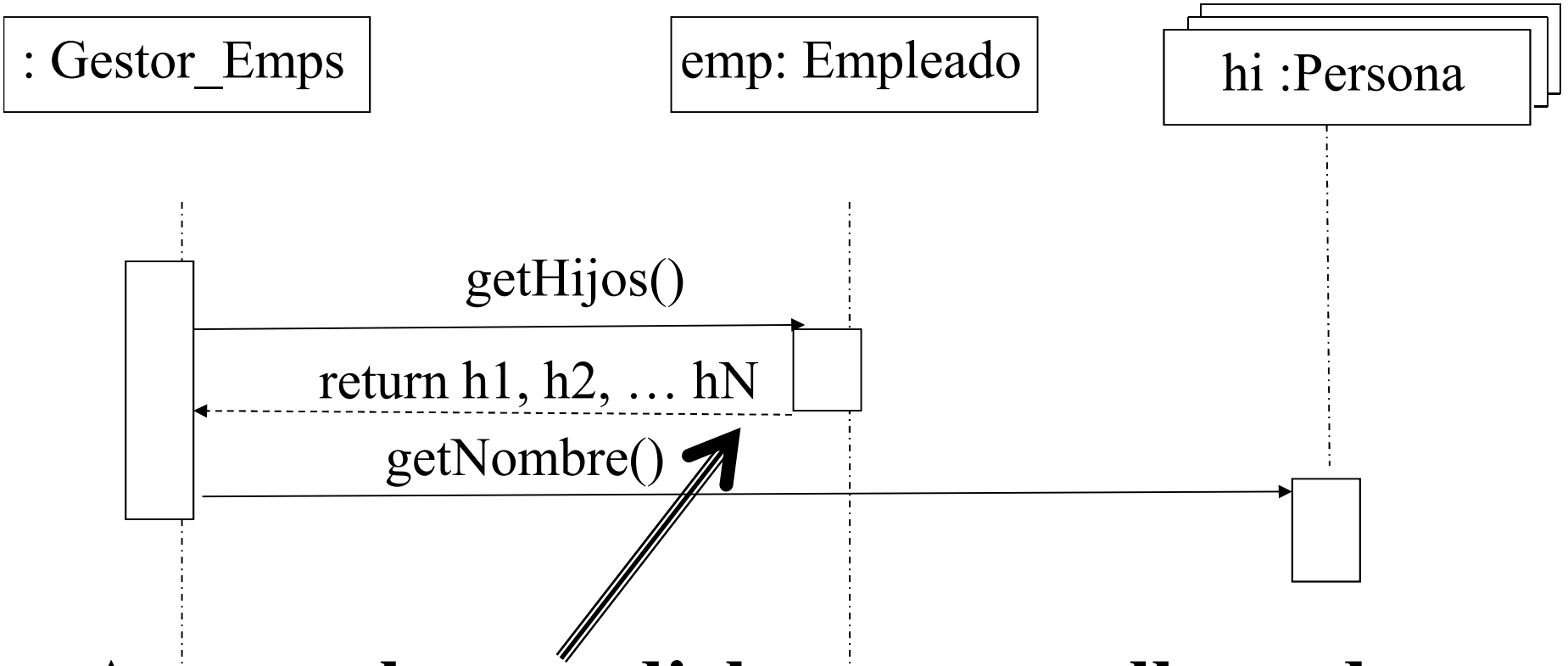

**Esta es otra manera de especificar una repetición: usando un rectángulo en el que se encuadran todos los pasos de mensajes que se van a repetir. Se puede indicar la cardinalidad de cuántas veces se repite o una condición de hasta cuándo se repite.**

# Diagrama de Secuencia en UML



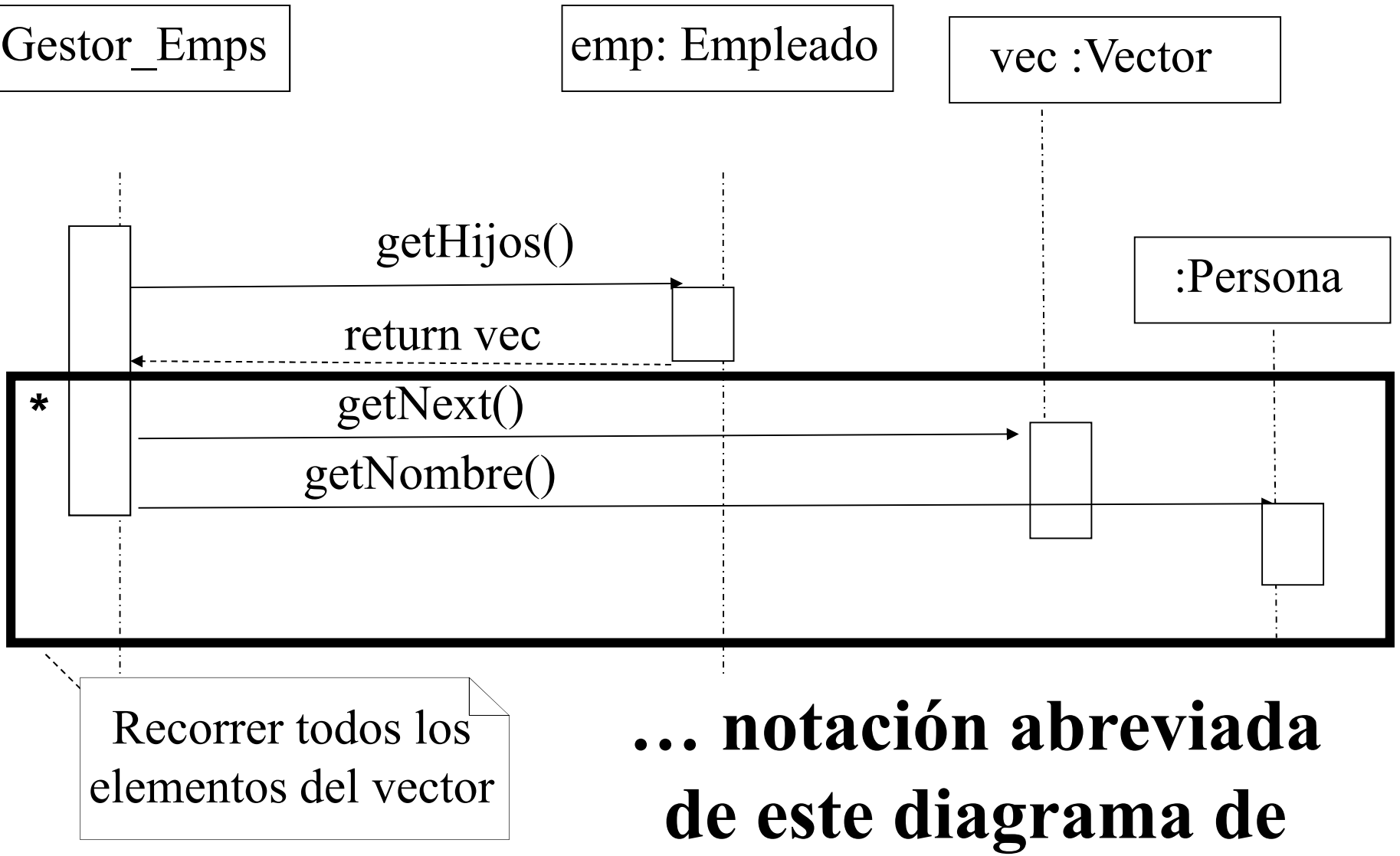

**Esta es otra manera de especificar una repetición: utilizando una nota UML**

# Diagrama de Secuencia en UML




**Aunque hemos dicho que una llamada a un método NO PUEDE DEVOLVER MÁS DE UN VALOR, permitiremos esto como una notación abreviada …**

# Diagrama de Secuencia en UML




**… notación abreviada de este diagrama de secuencia**

# Artefacto: Interfaz

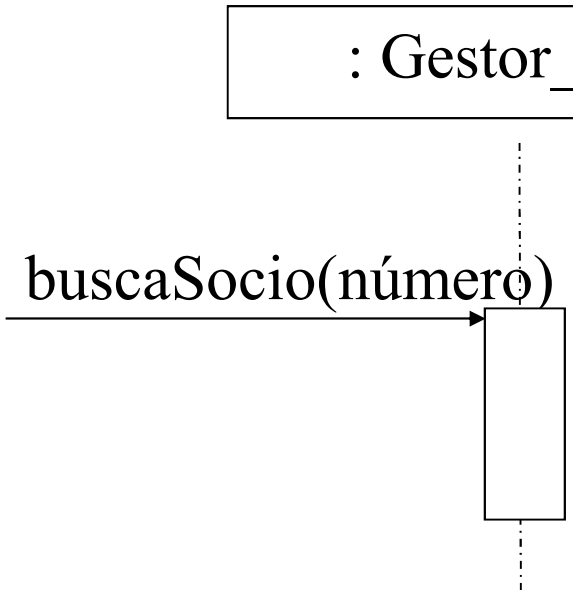
- Las interfaces se usan para especificar operaciones
- Separan la especificación de la funcionalidad de su implementación
- Las interfaces definen cómo se interactúa entre los subsistemas.
  - Las interfaces son requisitos para los equipos de desarrollo y sirven para que los distintos equipos puedan trabajar concurrentemente.

# Interfaz

- Supongamos que hay dos equipos de trabajo implicados en el SI de la biblioteca y que se han dividido el trabajo
  - Uno se encarga de los subsistemas SOCIOS y CATÁLOGO
  - Otro se encarga del subsistema PRÉSTAMOS
- Es evidente que cada equipo puede necesitar ciertos métodos del otro.
- Lo que tienen que hacer es definirlos utilizando interfaces
  - clases con sólo métodos no implementados

# Interfaz

- ¿Qué pueden necesitar del subsistema SOCIOS?
  - Un método que dado un número de socio devuelva una referencia al objeto socio.
  - Un método para añadir un nuevo socio al sistema.
  - Un método para …




**Es una INTERFAZ que el equipo de PRÉSTAMOS puede utilizar. El equipo de SOCIOS ya proporcionará una implementación**

# Artefacto: descripción de la arquitectura (diseño)

- Es una descripción que contiene la vista arquitectural del modelo del diseño
- Los siguientes artefactos son arquitecturalmente significativos:
  - Descomposición del modelo en subsistemas junto con sus interfaces
  - Clases de diseño claves
  - Realizaciones de casos de uso con funcionalidad crítica

# 4. Dos patrones de diseño para acceder a los datos

- 1) Usando un lenguaje/sistema OO para el almacenamiento de datos que permita trabajar con objetos “persistentes”
  - Es posible si se utiliza un SGBD OO
  - O si se usa Java + mecanismos de serialización de Java
- 2) Usando un SGBD relacional para el almacenamiento
  - Es posible si disponemos de un lenguaje con un API que permita conectarse con un SGBD, por ejemplo si se usa Java + JDBC

# Acceso a los datos usando un sistema OO

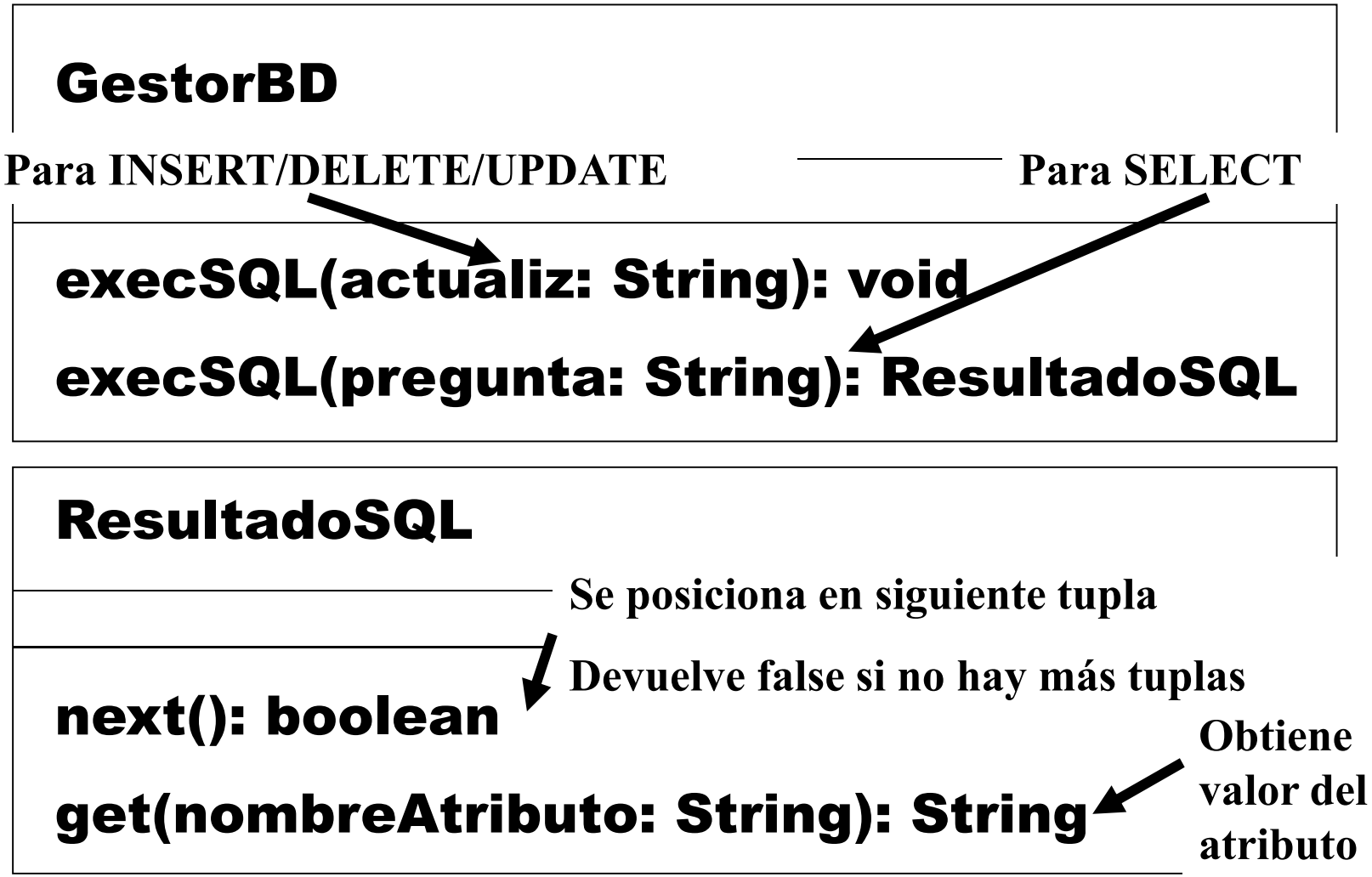
- El diseño es prácticamente equivalente al diagrama de colaboración de análisis
  - Usar nombres de métodos y no responsabilidades
  - Varias clases de diseño que sean interfaces gráficas
  - Detallar casos excepcionales

**Las clases de diseño que corresponden a las clases entidad pueden tener métodos**

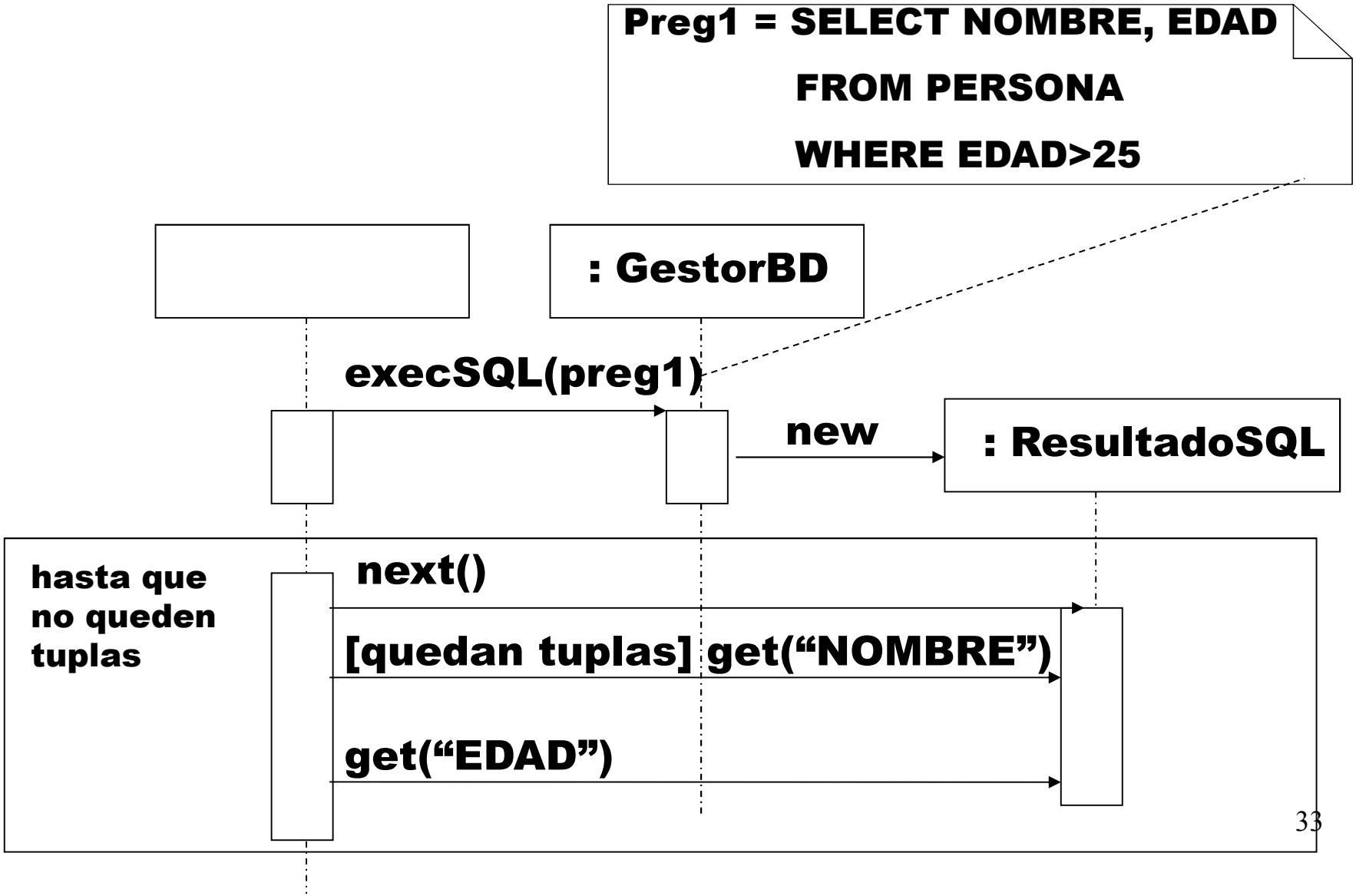
# Acceso a los datos usando un SGBD relacional

- El diseño cambia con respecto al diagrama de colaboración de análisis
  - MODELO DEL DOMINIO / CLASES ENTIDAD (OO)
  - ➔ ESQUEMA LÓGICO DE UNA BD RELACIONAL
    - Se trata en la asignatura DBD. Supondremos que se sabe ya.
  - CLASE DISEÑO (GESTOR BD) permite ejecutar SQL
    - SQL se trata en FBD y DBD. Supondremos que se sabe ya.
  - Las CLASES ENTIDAD NO pueden tener MÉTODOS

# Acceso a los datos usando un SGBD relacional



# Acceso a los datos usando un SGBD relacional



# Anexo: Artefacto: Modelo de despliegue

- Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los distintos nodos
  - Cada nodo representa un recurso computacional (procesador, dispositivo hardware,…).
  - Las relaciones entre nodos representan formas de comunicación entre ellos (internet, intranet, bus,…).
  - La funcionalidad de un nodo se define por los componentes desplegados en él.
- El modelo de despliegue es la correspondencia entre la arquitectura del software y la arquitectura del sistema

# Artefacto: descripción de la arquitectura (despliegue)

- Es una descripción que contiene la vista arquitectural del modelo del despliegue
- Todos los aspectos del modelo de despliegue deben mostrarse en la vista de la arquitectura

# Anexo: Trabajadores

- Arquitecto
  - Responsable de la integridad del modelo de diseño y de despliegue
- Ingeniero de casos de uso
  - Responsable de la integridad de los casos de uso
- Ingeniero de componentes
  - Define y mantiene las operaciones métodos, atributos, relaciones y requisitos de implementación de clases de diseño. Debe mantener la integridad de los subsistemas controlando que se realizan los interfaces.

# Anexo: Actividades en el FT de diseño

- 1.- Diseño de la arquitectura
  - Identificar nodos y configuraciones de red
  - Identificar subsistemas y sus interfaces
  - Identificar clases de diseño arquitecturalmente significantes
  - Identificar mecanismo de diseño genéricos
- 2.- Diseñar caso de uso
  - Identificar clases de diseño participantes
  - Describir interacciones de objetos de diseño
  - Identificar los subsistemas participantes e interfaces
  - Describir interacciones entre subsistemas
  - Capturar requisitos de implementación

# Actividades en el FT de diseño

- 3.- Diseñar clase
  - Perfilar la clase de diseño
  - Identificar las operaciones
  - Identificar los atributos
  - Identificar asociaciones y agregaciones
  - Identificar generalizaciones
  - Describir los métodos
  - Describir los estados
  - Tratar los requisitos especiales
- 4.- Diseñar subsistema
  - Mantener dependencias entre subsistemas
  - Mantener los interfaces proporcionados por el subsistema
  - Mantener los contenidos del subsistema