



Ingeniería de Software

ANÁLISIS Y DISEÑO DE
SISTEMAS CON



Auxiliar: Andrés Neyem
aneyem@dcc.uchile.cl
Oficina 418 de Doctorado

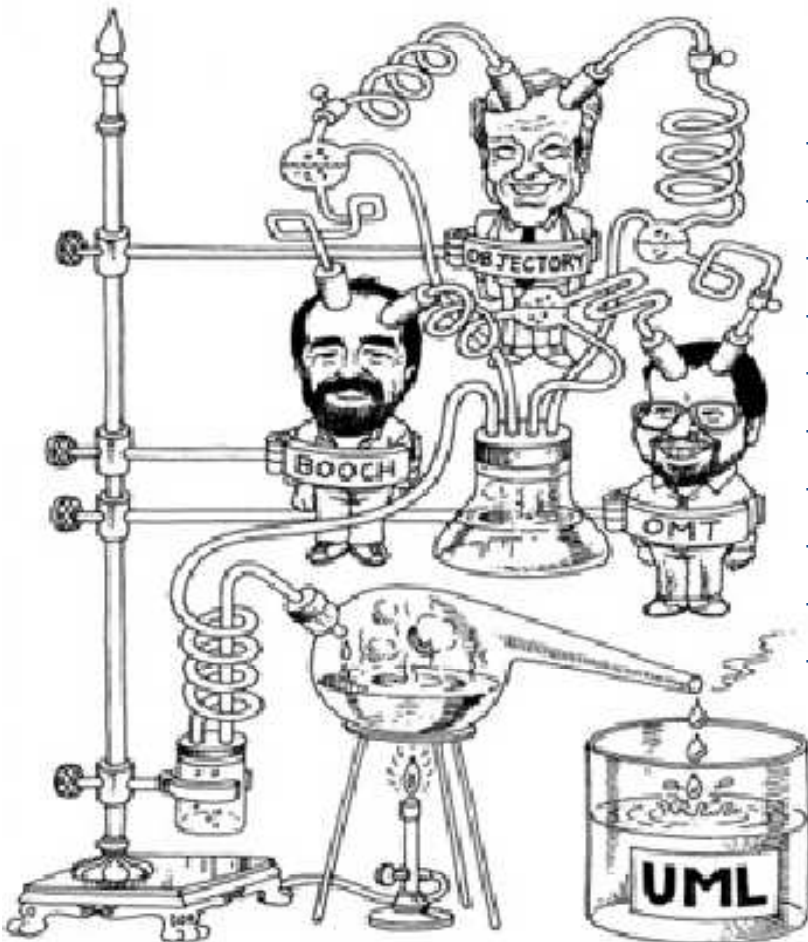
Auxiliar – 27 de Agosto de 2007



Repaso

Historia de los lenguajes de modelamiento OO

- 1994 Rumbaugh se une a Booch (vers. 0.8 de Unified Method) en Rational.
- 1995 presentan el Unified Method en OOPLSA.
- Jacobson se une a Rational (versión 0.9 de UML, junio de 1996).
- Versión 1.0 enero 1997, aprobación de OMG, Object Management Group.
- Versión 1.1 en noviembre de 1997 (con nuevas recomendaciones).
- Versión 1.2 en junio de 1998.
- Versión 1.3 a fines de 1998.
- Versión 1.4 en septiembre de 2001.
- Versión 1.5 en marzo de 2003 (versión oficial de OMG).
- Versión 2.0 en octubre de 2004.





Repaso

- Modelado de requerimientos
 - Diagrama de casos de uso
- Modelado de la estructura
 - Diagrama de clases
 - Diagrama de objetos
- Modelado de la interacción
 - Diagrama de secuencias
 - Diagrama de colaboraciones
- Modelado del comportamiento
 - Diagrama de estados
 - Diagrama de actividades
- Herramientas de diseño
 - Diagrama de componentes
 - Diagrama de despliegue
- Organización del modelo
 - Diagrama de paquetes



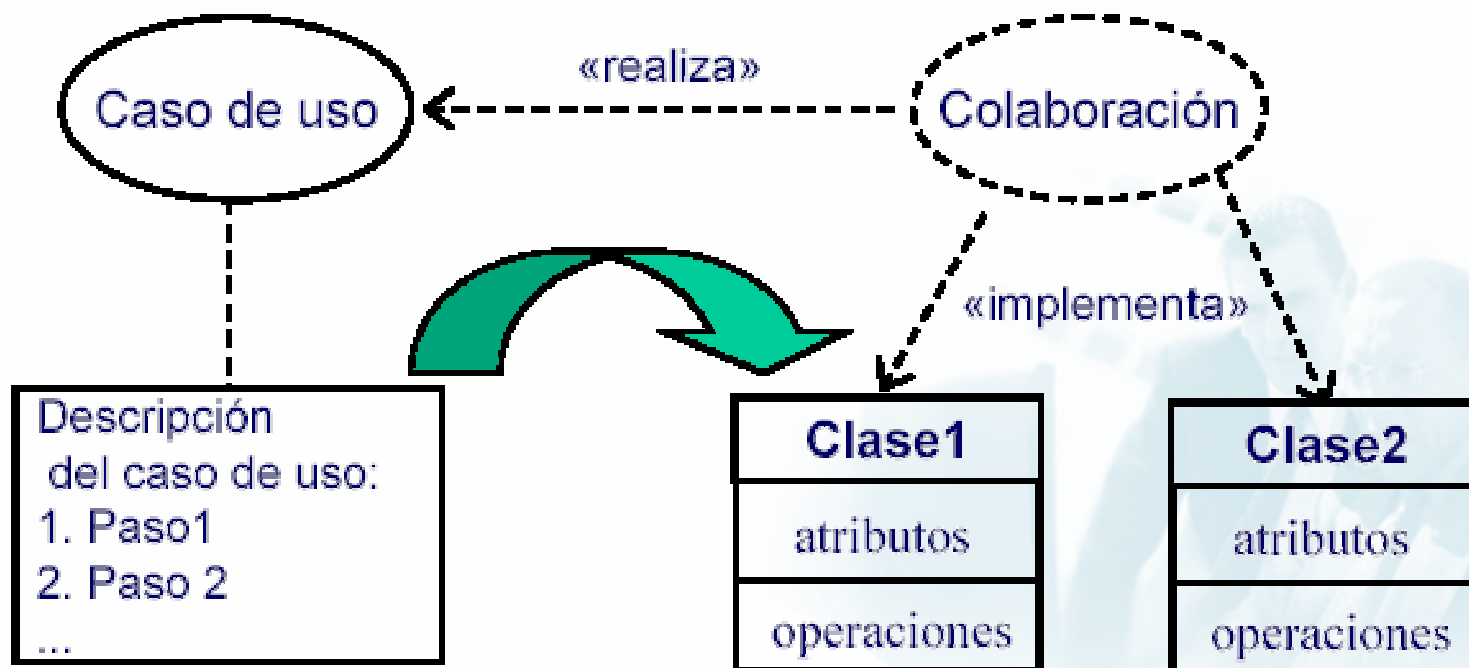
¿Qué es un Caso de Uso?

“Descripción de un conjunto de secuencias de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular”



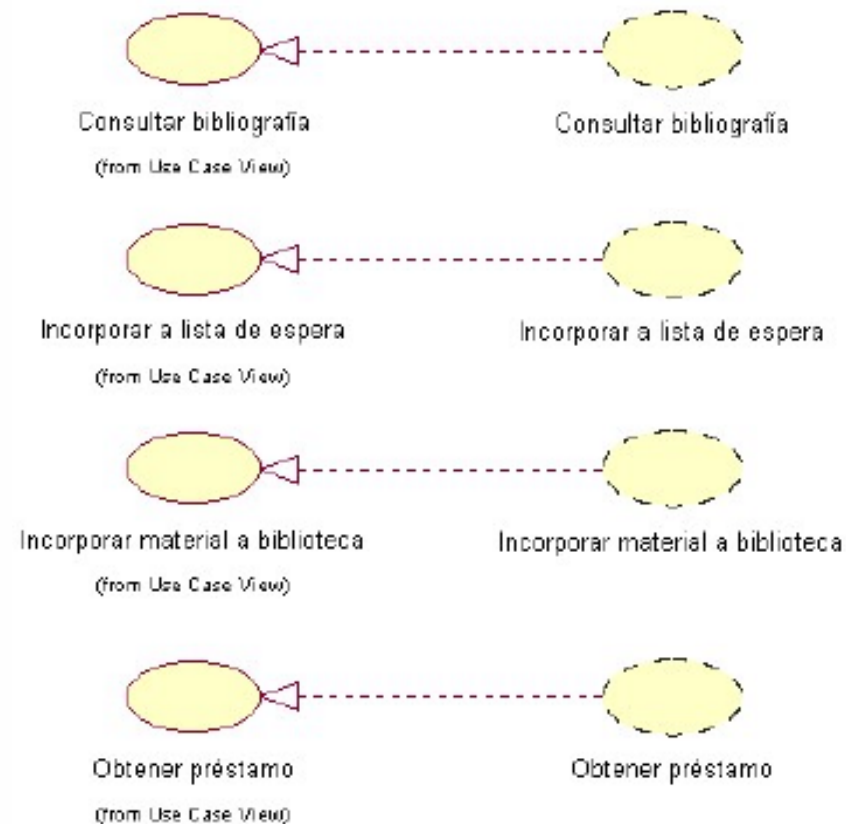
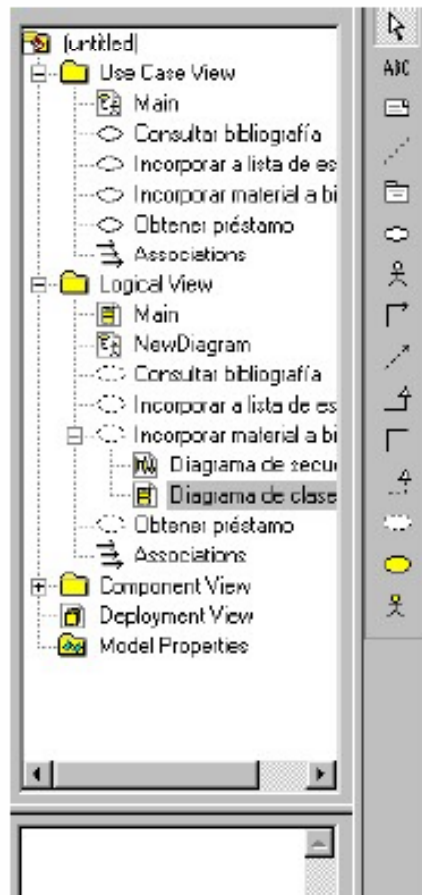
Realización de casos de uso

Las acciones descritas en un caso de uso deben ser ubicadas en objetos que colaboran entre sí para implementar la funcionalidad





Realización de casos de uso





Realización de casos de uso

- *Análisis textual de las operaciones*: las acciones contenidas en las descripciones de los casos de uso se pasan a operaciones en clases en una relación uno a uno
- *Construcción del modelo de clases*: permite identificar inicialmente las clases para asignar las operaciones





Diagrama de Secuencia

- Es uno de los dos diagramas de interacción que se propone en UML
- Describe la forma en la que colaboran entre sí los objetos para llevar a cabo sus respectivas responsabilidades
- Cada diagrama representa la funcionalidad de un único caso de uso
- Permite ver cómo se suceden cronológicamente los mensajes entre los objetos
- Proviene de los diagramas POSA de Buschmann
- Fueron utilizados por los tres autores del UML en sus respectivos métodos previos

Diagrama de Secuencia

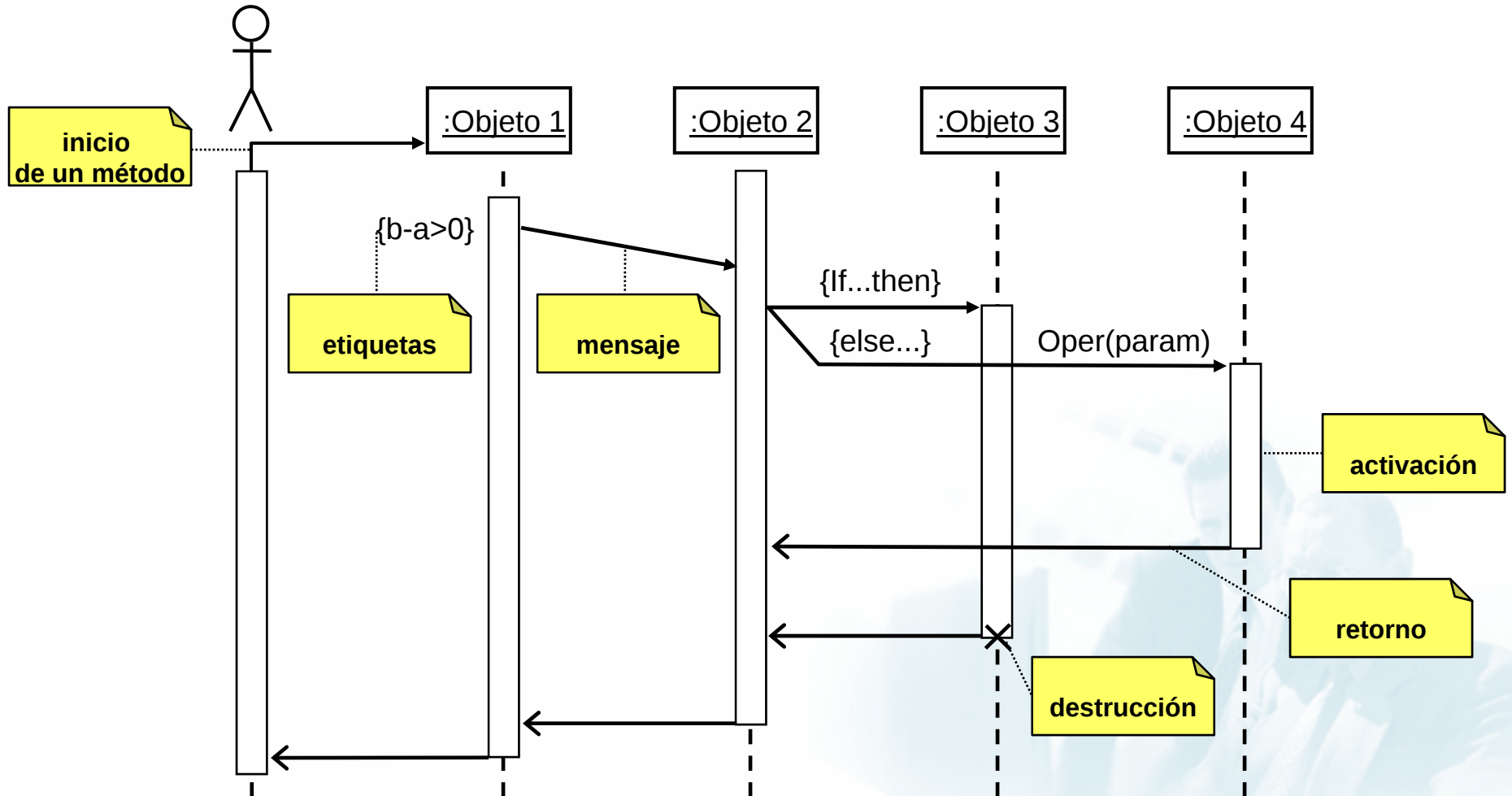




Diagrama de Secuencia

Mensajes



mensaje síncrono



mensaje asíncrono



*mensaje síncrono con respuesta
inmediata*



mensaje simple

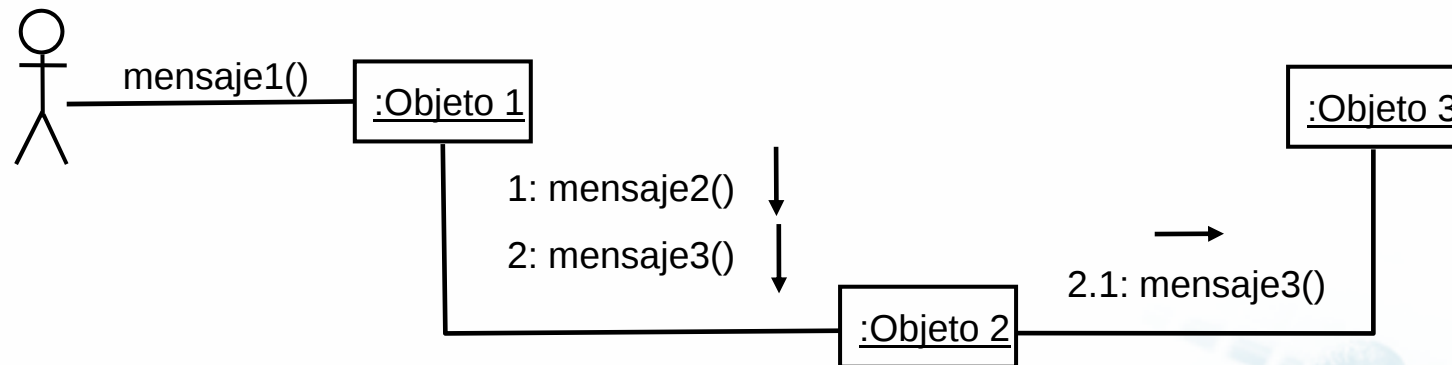


Diagrama de Colaboración

- Es otro de los diagramas de interacción que se incluye en UML
- No permite observar gráficamente la cronología de los mensajes
- Facilita la organización de los objetos en paquetes
- Destaca la conexión estática entre los objetos
- Mientras el diagrama de secuencias pone énfasis en el tiempo, el de colaboración lo hace en el espacio



Diagrama de Colaboración



“ilustran la interacciones entre objetos en un formato de grafo o red, en el cual los objetos se pueden colocar en cualquier lugar del diagrama”

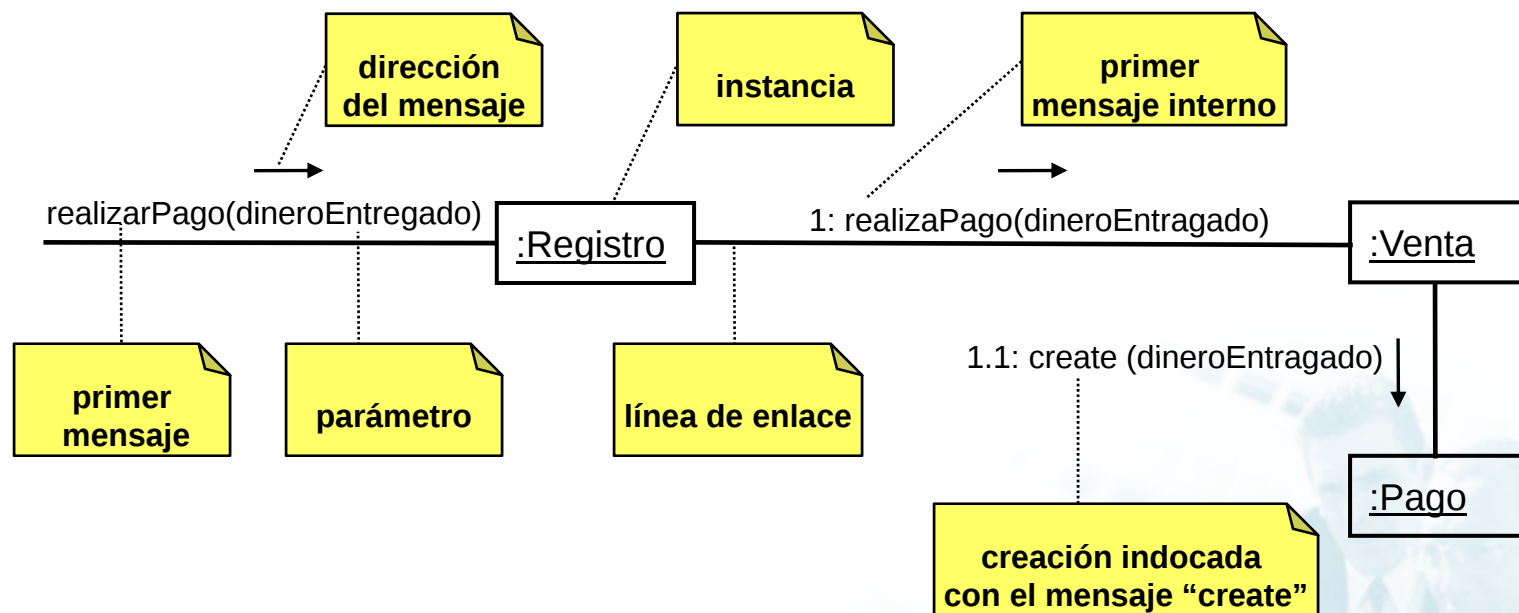


Diagramas de Interacción

Tipo	Puntos fuertes	Puntos débiles
Secuencia	Muestra claramente la secuencia u ordenación en el tiempo de los mensajes Notación simple	Fuerza a extender por la derecha cuando se añaden nuevos objetos; consume espacio horizontal.
Colaboración	Economiza espacio, flexibilidad al añadir nuevos objetos en dos dimensiones Es mejor para ilustrar bifurcaciones complejas, iteraciones y comportamientos concurrente	Difícil ver la secuencia de mensajes Notación más compleja

Diagramas de Interacción

Ejemplo de diagrama de colaboración: Realizar Pago



El diagrama se lee como sigue:

1. Se envía el mensaje de *realizarPago* a una instancia de Registro. No se identifica al emisor.
2. La instancia de Registro envía el mensaje *realizarPago* a una instancia de Venta.
3. La instancia de Venta crea una instancia de Pago.

Diagramas de Interacción

Ejemplo de diagrama de secuencia: Realizar Pago

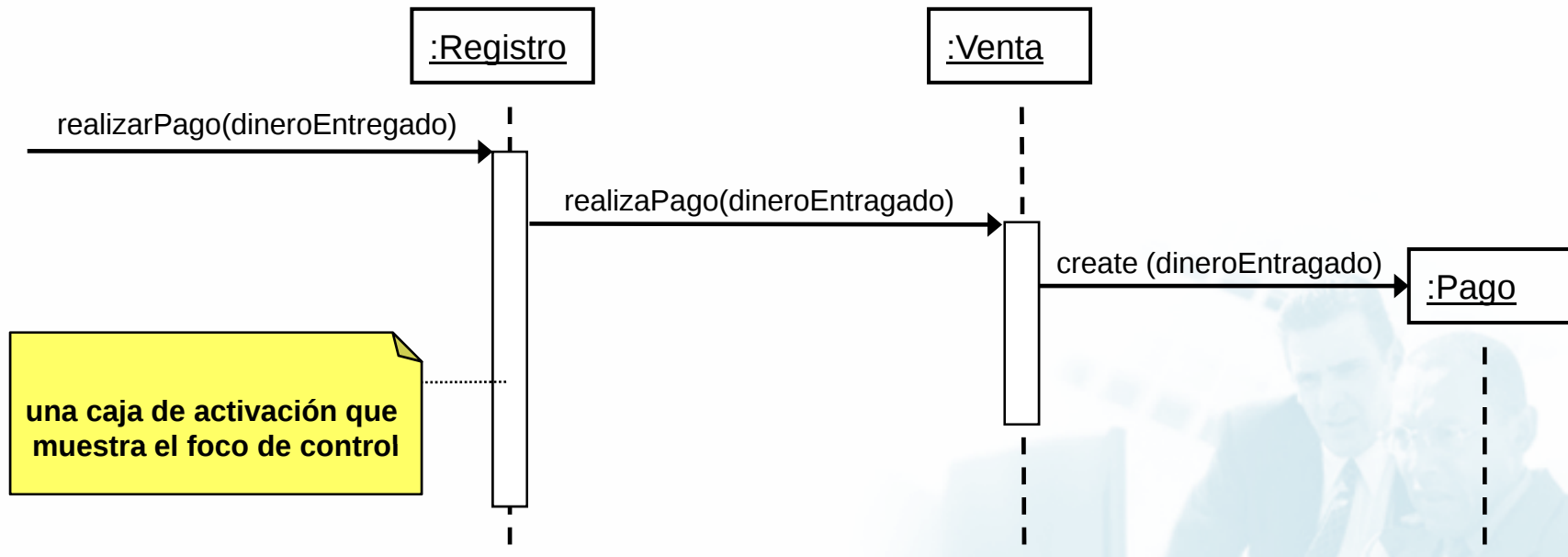




Diagrama de Estados

- Describe los estados posibles en la vida de los objetos
- Permite observar cómo cambian de estado los objetos a medida que ocurren los eventos
- Cada diagrama se utiliza para representar el ciclo de vida de los objetos de una única clase
- Proviene de las cartas de estado de David Harel
- Los emplearon Rumbaugh en OMT, Booch en su libro de 1994 y Jacobson con la incorporación de una vasta notación



Diagrama de Estados

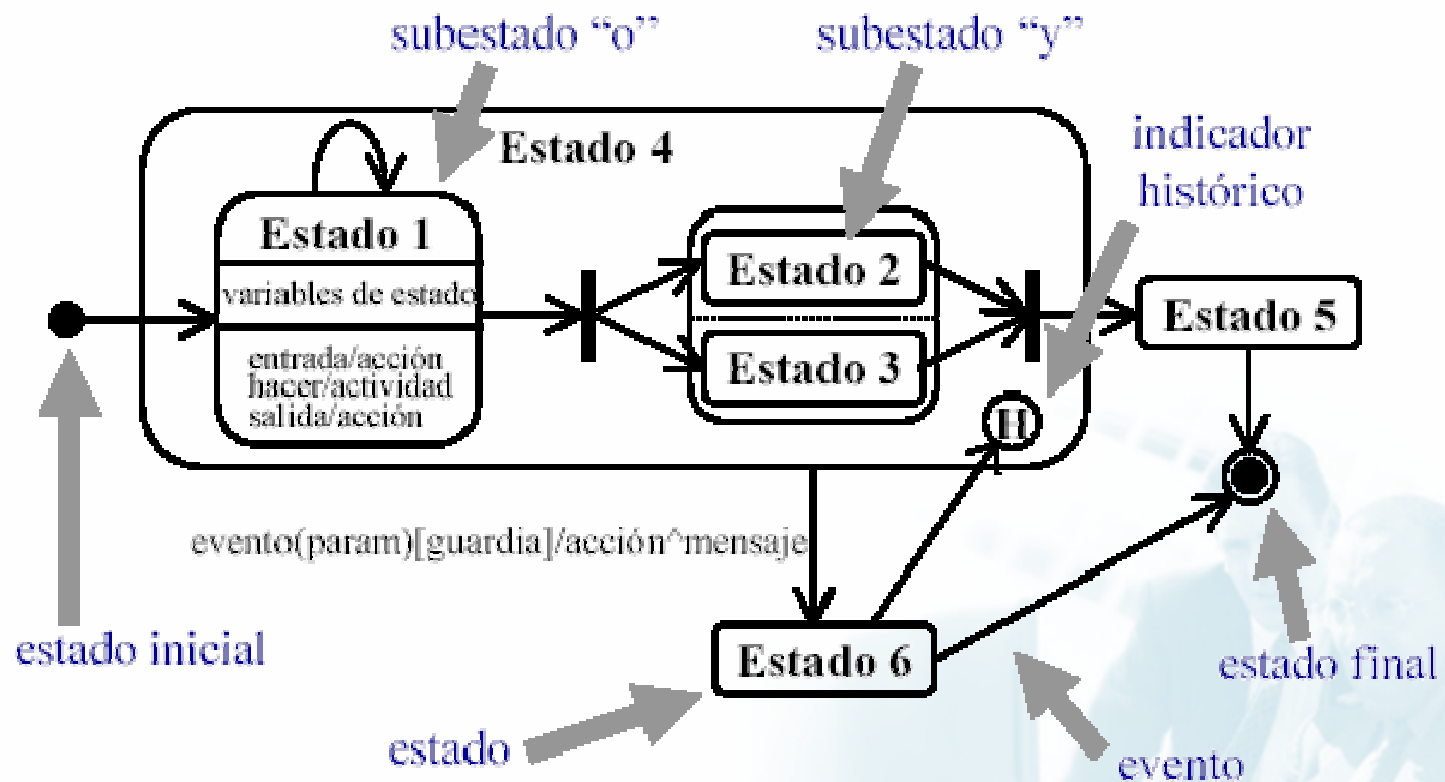




Diagrama de Estados

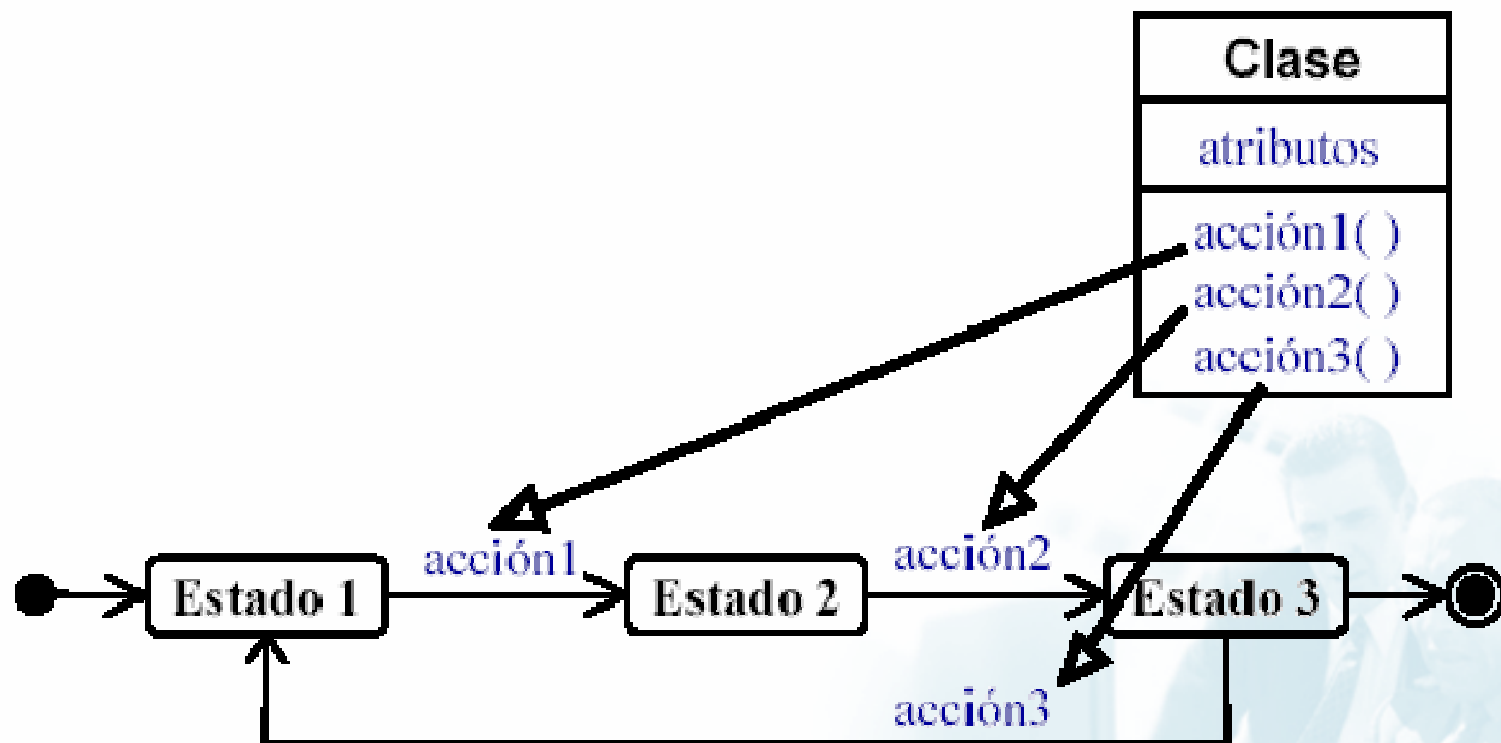




Diagrama de Actividades

- No posee antecedentes claros entre las herramientas de los autores de UML en sus propios métodos
- Proviene de varias técnicas, como diagramas de eventos de Odell y redes de Petri
- Permite destacar y sincronizar las operaciones concurrentes y establecer caminos alternativos
- Muestra el comportamiento combinado de varias clases, aunque éstas no se identifican si no se lo hace explícitamente
- Al igual que los diagramas de estado, se emplea para describir comportamientos complejos

Diagrama de Actividades

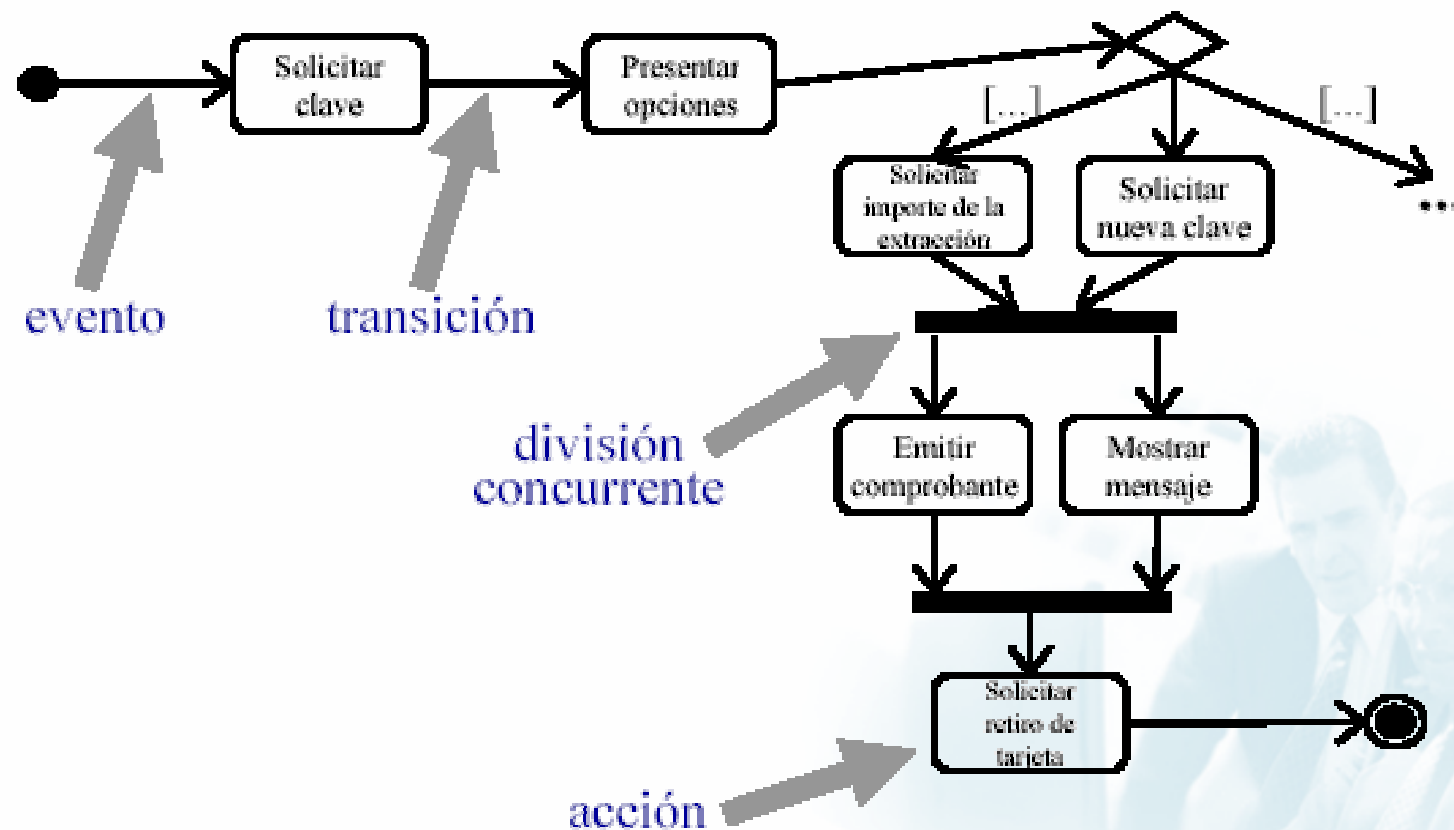


Diagrama de Actividades

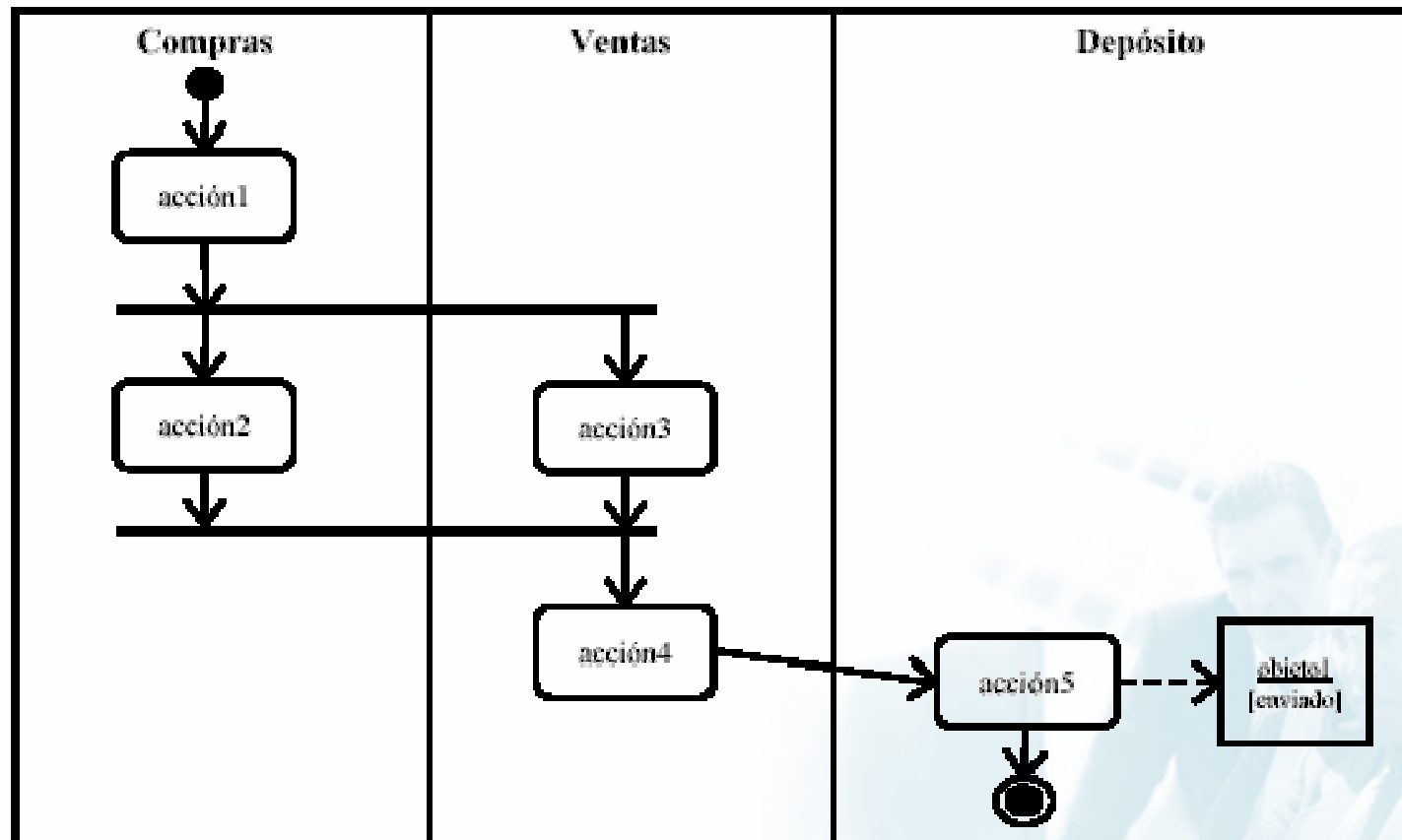


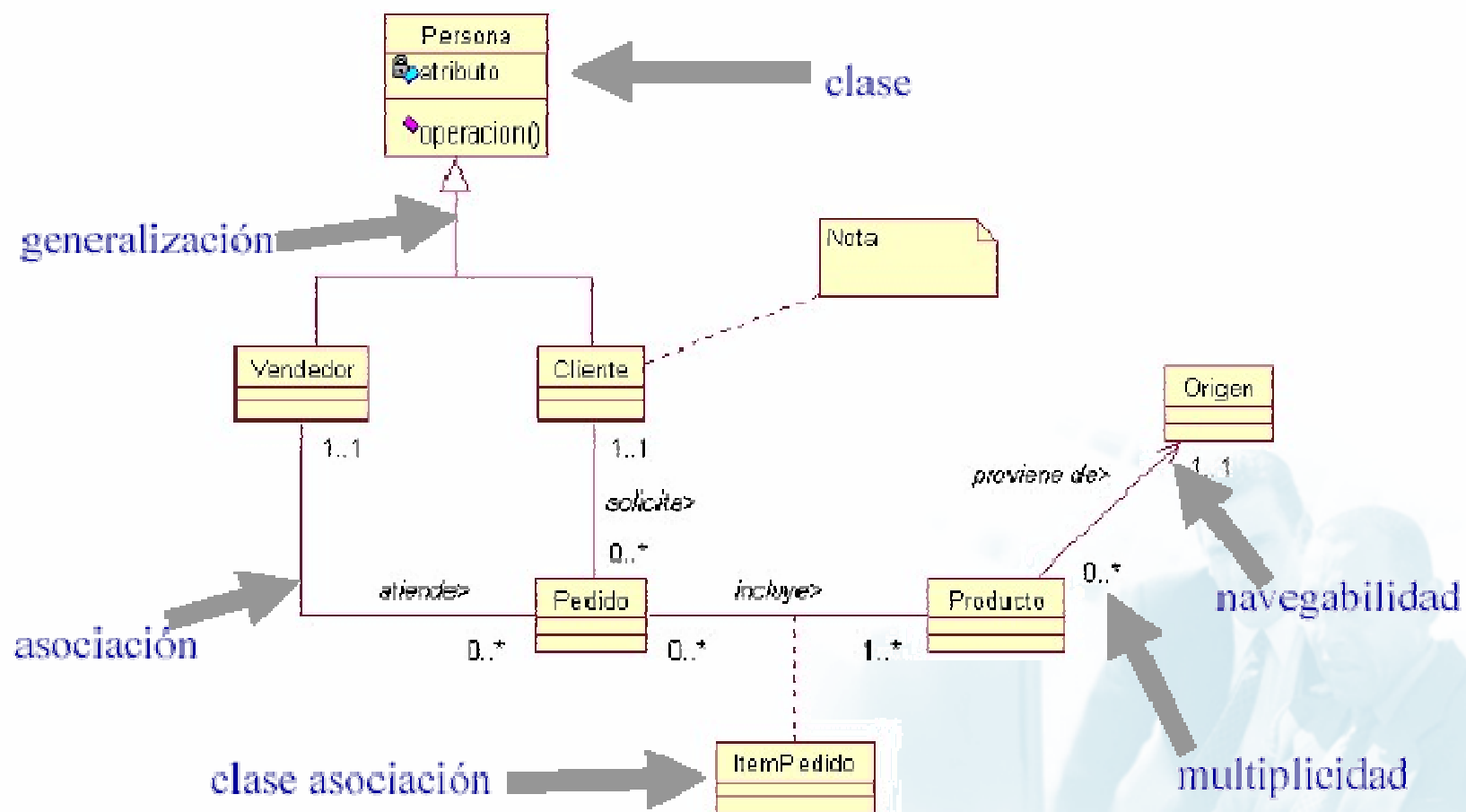


Diagrama de Clases

- Proviene de los diagramas de entidad-relación de Chen ('70)
- Fueron extendidos con conceptos como generalización y agregación ('80)
- Incorporados por los autores orientados a las características de los objetos
- Permiten modelar la estructura estática de los sistemas
- Utilizados en UML para la construcción de los metamodelos
- Aunque también fueron empleados por Booch, conservan el aspecto de la notación propuesta por Rumbaugh en OMT



Diagrama de Clases



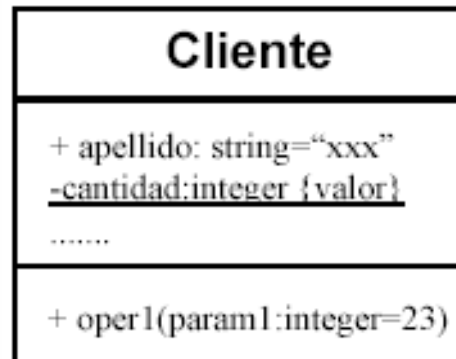


Objetos y clases

- **Clase:** abstracción de un conjunto de objetos que poseen características, comportamientos, relaciones y semántica semejantes.
- **Objeto:** entidad existente en el mundo real que se distingue del resto por sus características, comportamientos, relaciones y semántica.



Diagrama de Clases



■ *Atributos:*

[visibilidad] [/] nombre [multiplicidad] [:tipo] [= valor inicial] [{propiedades}]

■ *Operaciones:*

[visibilidad] nombre [(lista de parámetros)] [:tipo de respuesta] [{propiedades}]

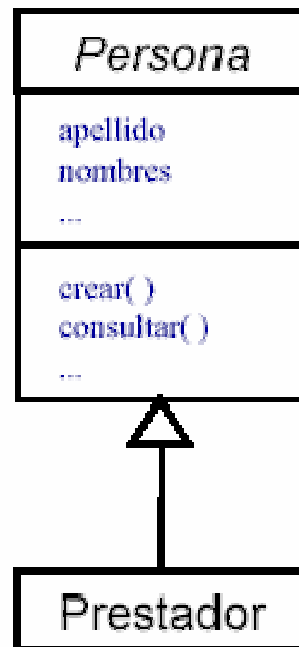
■ *Visibilidad:*

existe definición a nivel público (+), privado (-) y protegido (#)



Diagrama de Clases

- **Generalización o herencia:** relación jerárquica entre clases en la que una clase hereda todos los miembros de otra más general (relación “tipo-de”)



```
objPrestador.crear()
objPrestador.apellido = "Pérez"
```



Diagrama de Clases

- *Asociaciones n-arias ($n > 2$)*

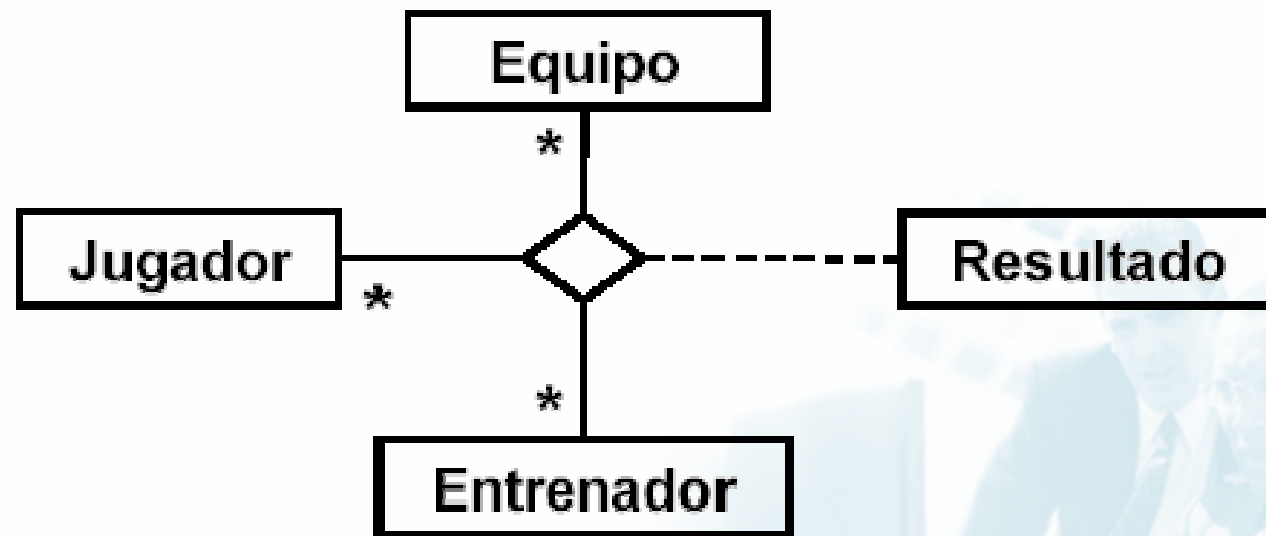




Diagrama de Clases

- **Agregación:** relación jerárquica entre objetos en la que uno es el todo y los otros son las partes.
 - **Agregación simple (rombo sin relleno):** relación todo-parte, contenedor-contenido, conjunto-elemento. Implica que la parte podría estar en muchas instancias compuestas.
 - **Agregación de composición (rombo con relleno negro):** agregación en la que las partes nacen y mueren con el todo. Significa que la parte es un miembro de un único objeto compuesto y que existe una dependencia de existencia.

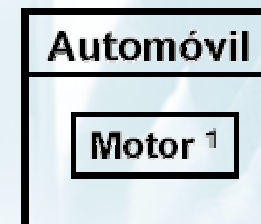
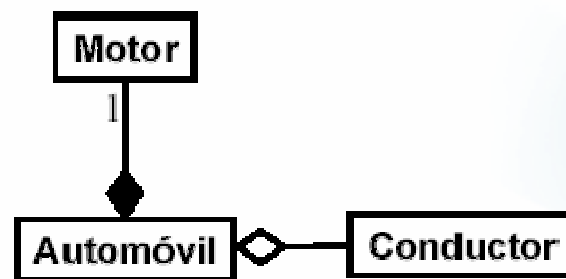




Diagrama de Clases

- **Interfaz:** clase con declaración de operaciones, sin implementación y sin atributos.

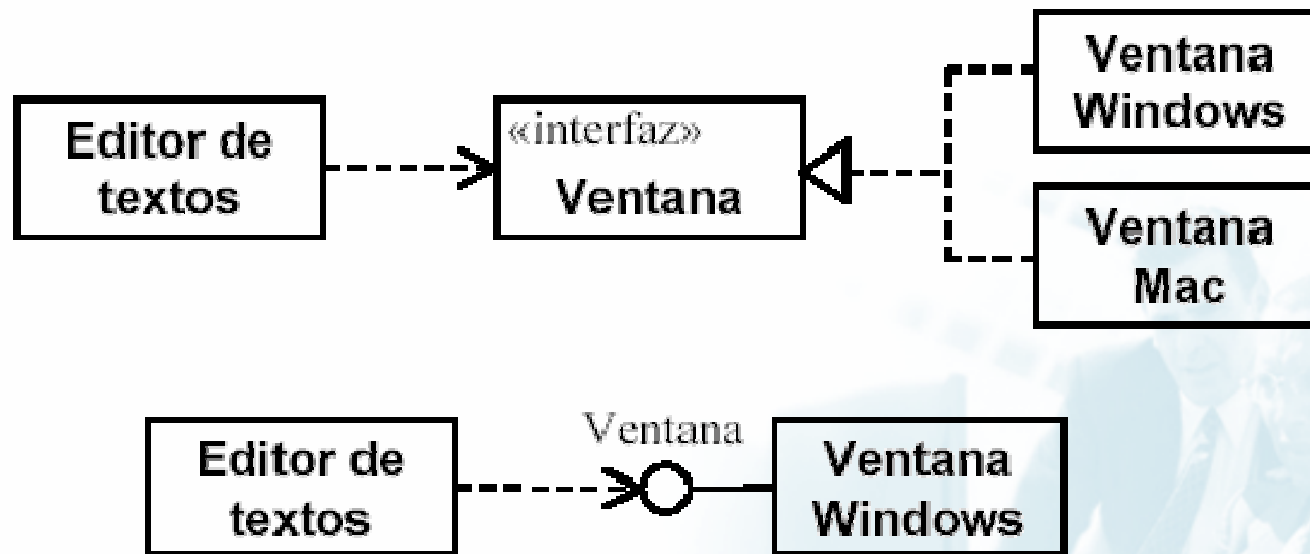




Diagrama de Clases

- **Asociación cualificada:** consiste en indicar la necesidad de una estructura de datos, estilo “diccionario”, en un extremo de una asociación. Se emplea en multiplicidades $1..*$ y $*..*$ a fin de reducirlas





Diagrama de Clases

- **Clase asociación:** es una asociación que se modela como clase o viceversa (Importante: la clase asociación tiene multiplicidad 1..1 con la asociación).

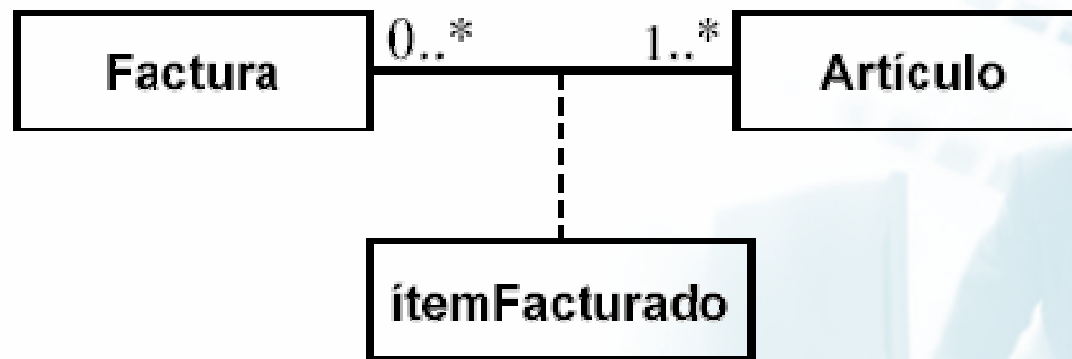




Diagrama de Clases

- **Clase parametrizada:** (“template”) es la descripción de una clase con uno o más parámetros.

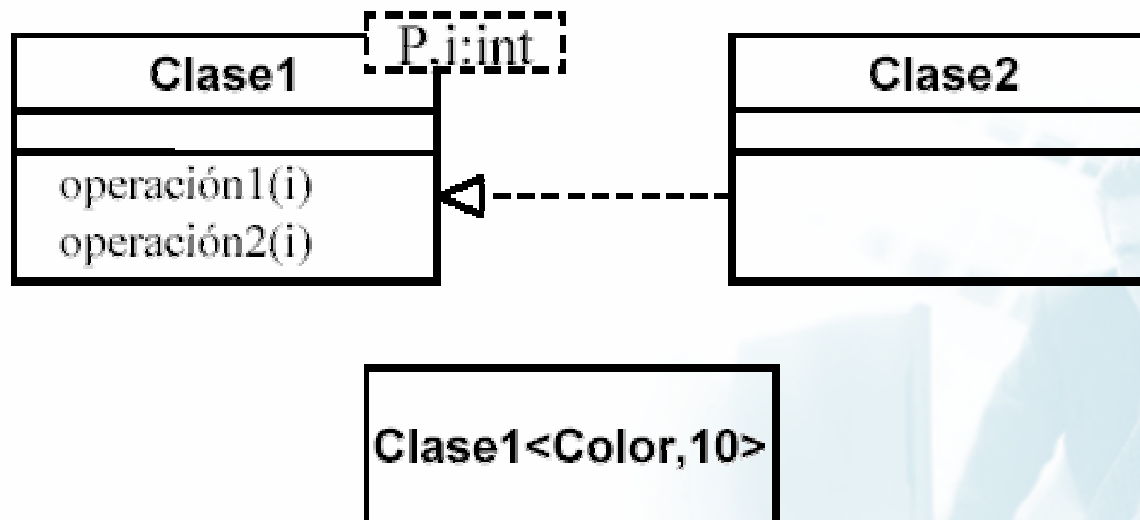




Diagrama de Clases

Pasos recomendados:

- elaborar una lista de clases candidatas
- detectar clases con diferentes niveles de abstracción
- definir las clases y colocarles sus atributos y comportamientos
- elegir la clase más representativa y colocarla en el centro del modelo
- asociar una a una el resto de las clases
- determinar multiplicidad y condicionalidad
- incorporar clases asociativas
- incorporar agregación y generalización
- verificar y validar el modelo contra los requerimientos





Diagrama de paquetes

- Permite administrar la complejidad del sistema al subdividirlo en porciones de menor tamaño
- Corresponde a las categorías del método de Booch
- Se pueden aplicar a diferentes elementos de modelado, no sólo a clases
- Permite establecer las dependencias entre paquetes (que no son de carácter transitivo) a fin de reducirlas
- También permite reducir los bucles de dependencias

Diagrama de paquetes

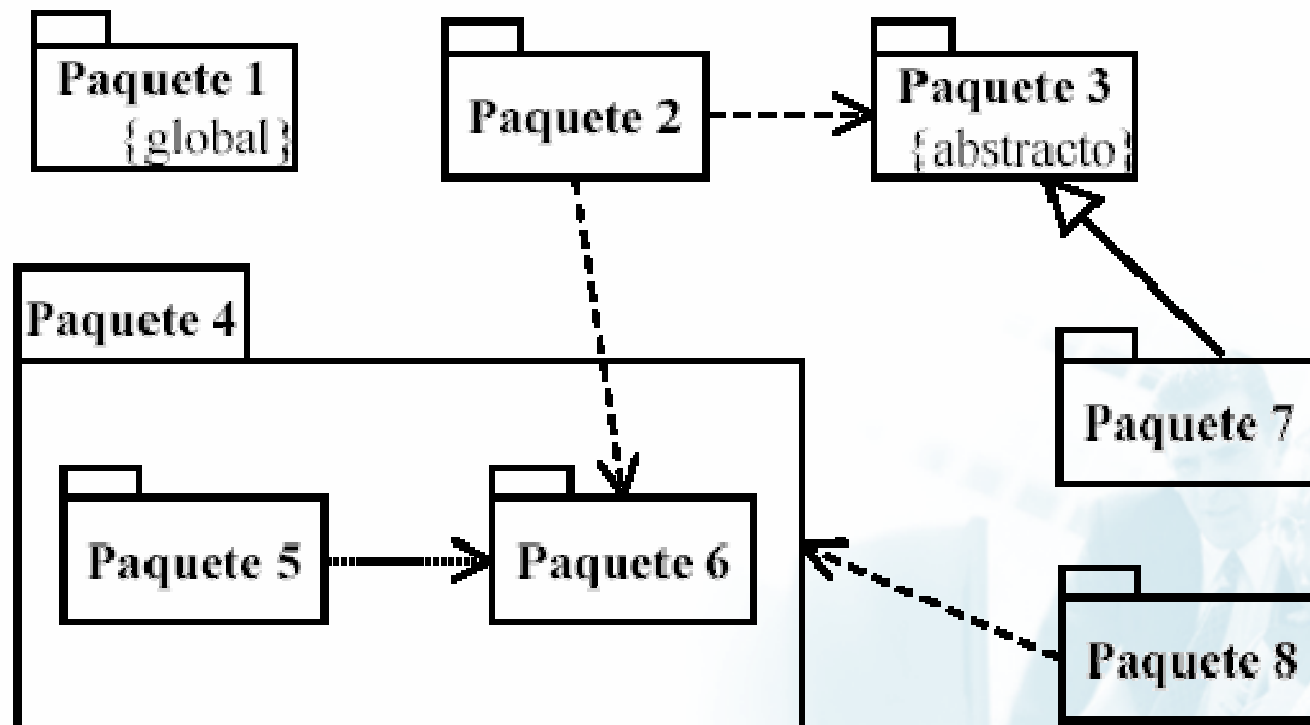


Diagrama de paquetes

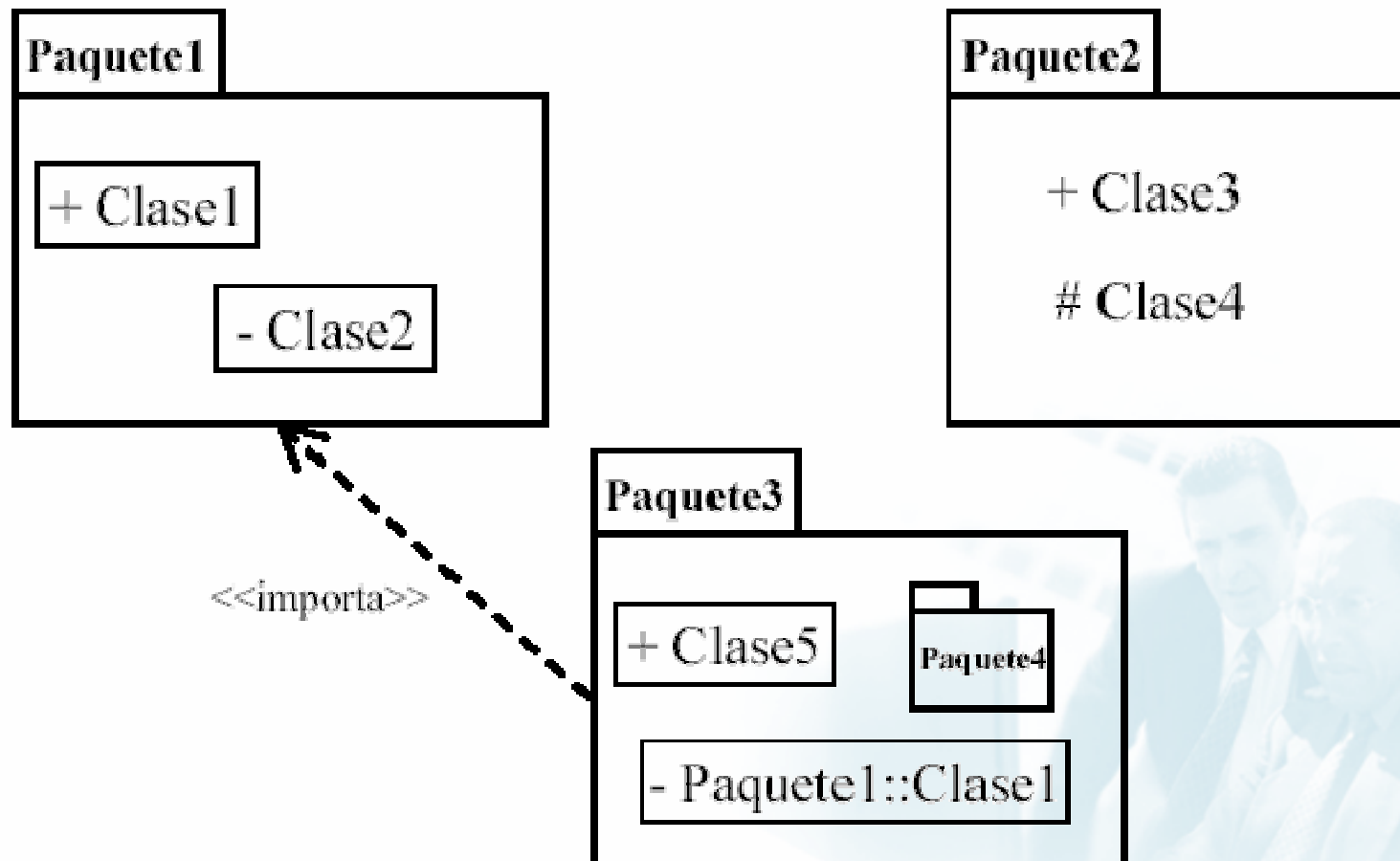




Diagrama de Componentes

- Este diagrama, junto al de despliegue, corresponde al grupo de herramientas de implementación de UML
- Representa módulos físicos de código
- Es importante que cada componente sea equivalente a un paquete
- De esta manera, las dependencias entre componentes con las mismas que las existentes entre los paquetes



Diagrama de Componentes

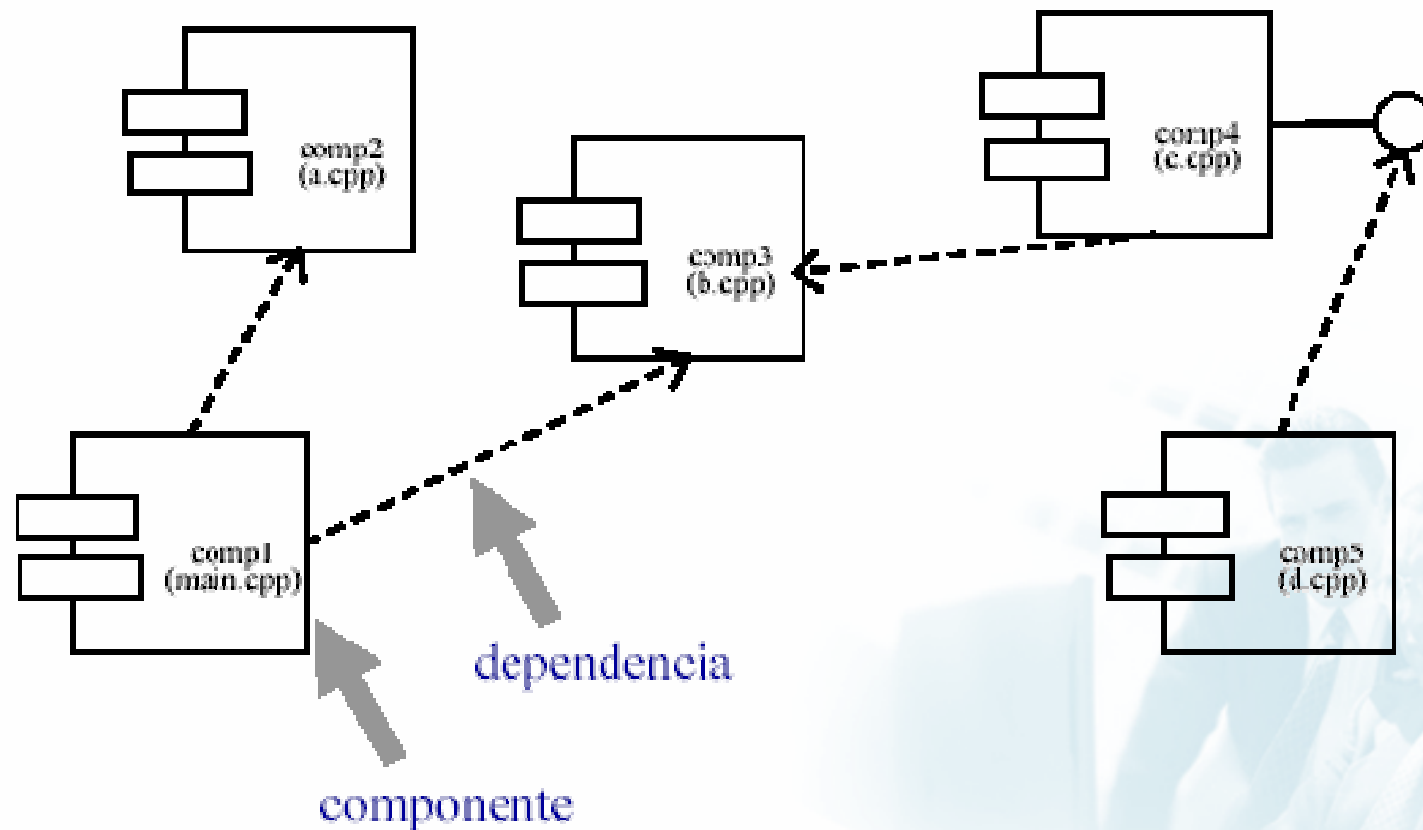




Diagrama de Componentes

- ***Existen tres tipos de componentes:***
 - de compilación (código fuente)
 - de linkeditado (archivos binarios, librerías estáticas)
 - de ejecución (ejecutables, tablas de BD, librerías dinámicas)
- ***Los estereotipos básicos son:***
 - «**file**»: código fuente o datos
 - «**page**»: página Web
 - «**document**»: texto, imágenes
 - «**executable**»: puede ejecutarse en un nodo
 - «**library**»: librería estática o dinámica
 - «**table**»: tabla de base de datos



Diagrama de Despliegue

- Es la segunda herramienta de implementación de UML
- Muestra las relaciones entre los componentes de hardware y software del sistema
- Permite observar dónde se encuentran físicamente los paquetes en el sistema
- La notación gráfica también proviene de Booch

Diagrama de Despliegue

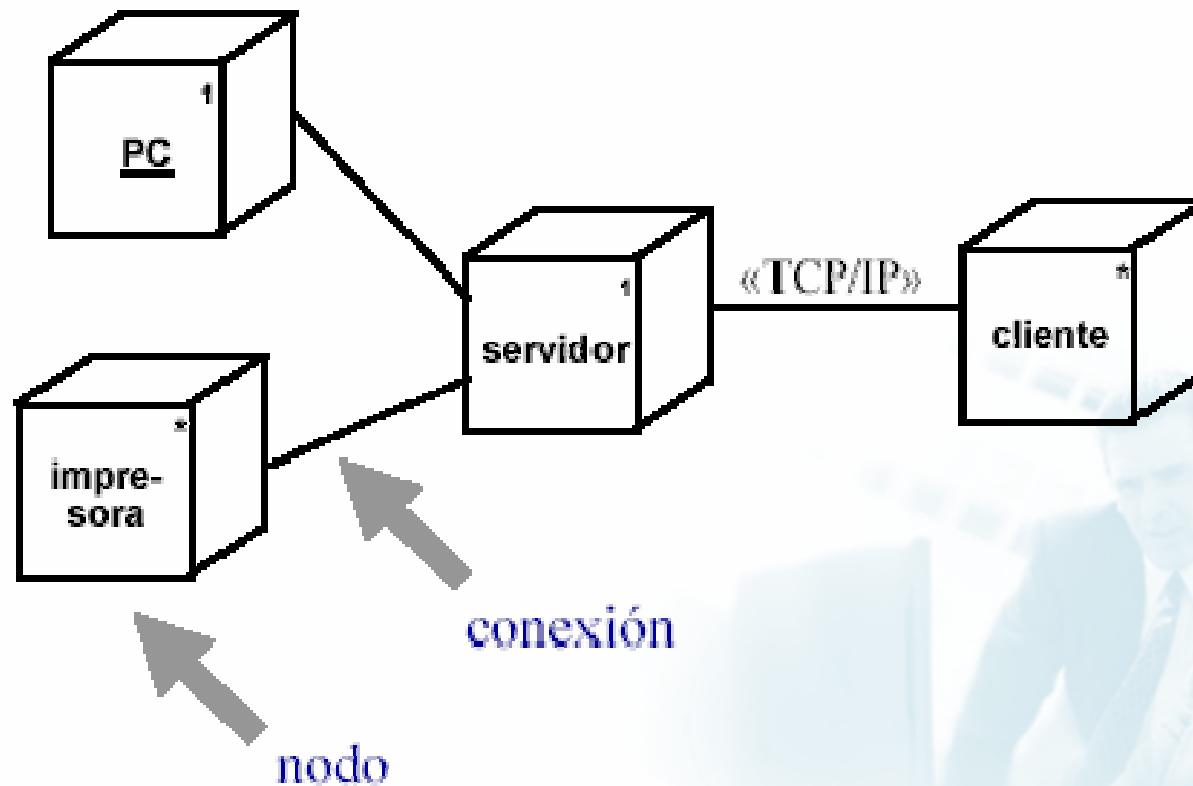


Diagrama de Despliegue

