

# Tema 5: Técnicas de Especificación y Modelado

BLOQUE II: ESPECIFICACIÓN DE REQUISITOS  
Y ANÁLISIS DE LOS SISTEMAS SOFTWARE

Ingeniería del Software

Grado en Ingeniería Informática

Curso 2024/2025



# Índice

1. Introducción al modelado de software y UML
  1. El rol de los modelos en la ingeniería del software
  2. Técnicas orientadas a objeto: Lenguaje Unificado de Modelado
  3. El modelo conceptual de UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
  4. Diagrama de Interacción
    - Diagramas de Secuencia
    - Diagramas de Colaboración
3. El Modelo Estructural en UML
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. Diagrama de Paquetes
  4. Diagrama de Componentes
  5. Diagrama de Despliegue
4. Técnicas Estructuradas
  1. Clasificación según el Enfoque de Representación
  2. Clasificación según el Enfoque de Modelado



# Índice

1. Introducción al modelado de software y UML
  1. El rol de los modelos en la ingeniería del software
  2. Técnicas orientadas a objeto: Lenguaje Unificado de Modelado
  3. El modelo conceptual de UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# Introducción al modelado de software y UML

## El rol de los modelos en la ingeniería del software

- “Los modelos son los planos del software”
  - Permiten razonar sobre el sistema antes de programar
  - Describen aspectos clave como estructura, comportamiento y despliegue
  - Facilitan la abstracción del sistema completo
  - Se usan para analizar el sistema en todas las etapas (antes, durante y después de su construcción)
  - Sustituyen documentos ambiguos y mejoran la precisión
  - Cumplen criterios (Stachowiak, 1973):
    - Representan un objeto o fenómeno real
    - Son un “espejo” (aunque parcial) de la realidad
    - Pueden reemplazar al original para algún propósito

### Fuente (lectura recomendada):

A. Vallecello. “Los planos del software”. Crónicas del Intangible (Blog El País). 22/11/2016.  
[https://elpais.com/tecnologia/2016/11/22/actualidad/1479834209\\_075442.html](https://elpais.com/tecnologia/2016/11/22/actualidad/1479834209_075442.html)



# Introducción al modelado de software y UML

## El rol de los modelos en la ingeniería del software

- Ingeniería de Sistemas Basada en Modelos (MBSE)
  - Uso formal de modelos para desarrollar sistemas
  - Cubre desde el análisis de requisitos hasta el sistema en funcionamiento
  - Los modelos son el principal medio de intercambio de información, superando los documentos
- Ingeniería del Software Guiada por Modelos (MDE)
  - Metodología de desarrollo basada en modelos de dominio
  - Enfocada en la abstracción del conocimiento y actividades del dominio
  - Aumenta productividad, fomenta la reutilización y simplifica el diseño
  - Permite generar código (model-driven development) y probarlo (model-based testing)



# Índice

1. Introducción al modelado de software y UML
  1. El rol de los modelos en la ingeniería del software
  2. Técnicas orientadas a objeto: Lenguaje Unificado de Modelado
  3. El modelo conceptual de UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

- A mediados de los noventa existían muchos métodos de análisis y diseño orientados a objetos:
  - Mismos conceptos con distinta notación
  - Mucha confusión
  - “Guerra de los métodos”
- En 1994, Booch, Rumbaugh (OMT) y Jacobson (Objectory/OOSE) deciden unificar sus métodos:

### Unified Modeling Language (UML)

- Proceso de estandarización promovido por el OMG (Object Management Group)



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

### Raíces Técnicas de UML

- OMT - *Object Modeling Technique* (Rumbaugh et al.)
  - Enfocado en análisis de datos para sistemas de información
  - Usa diagramas entidad-relación extendidos
- Método-Booch (G. Booch)
  - Ideal para sistemas concurrentes y de tiempo real
  - Relacionado con lenguajes como Ada
- OOSE - *Object-Oriented Software Engineering* (I. Jacobson)
  - Desarrollo basado en casos de uso
  - Fuerte en Ingeniería de Requisitos y sistemas de telecomunicaciones
- UML unifica estos conceptos y añade nuevos elementos



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

### Evolución

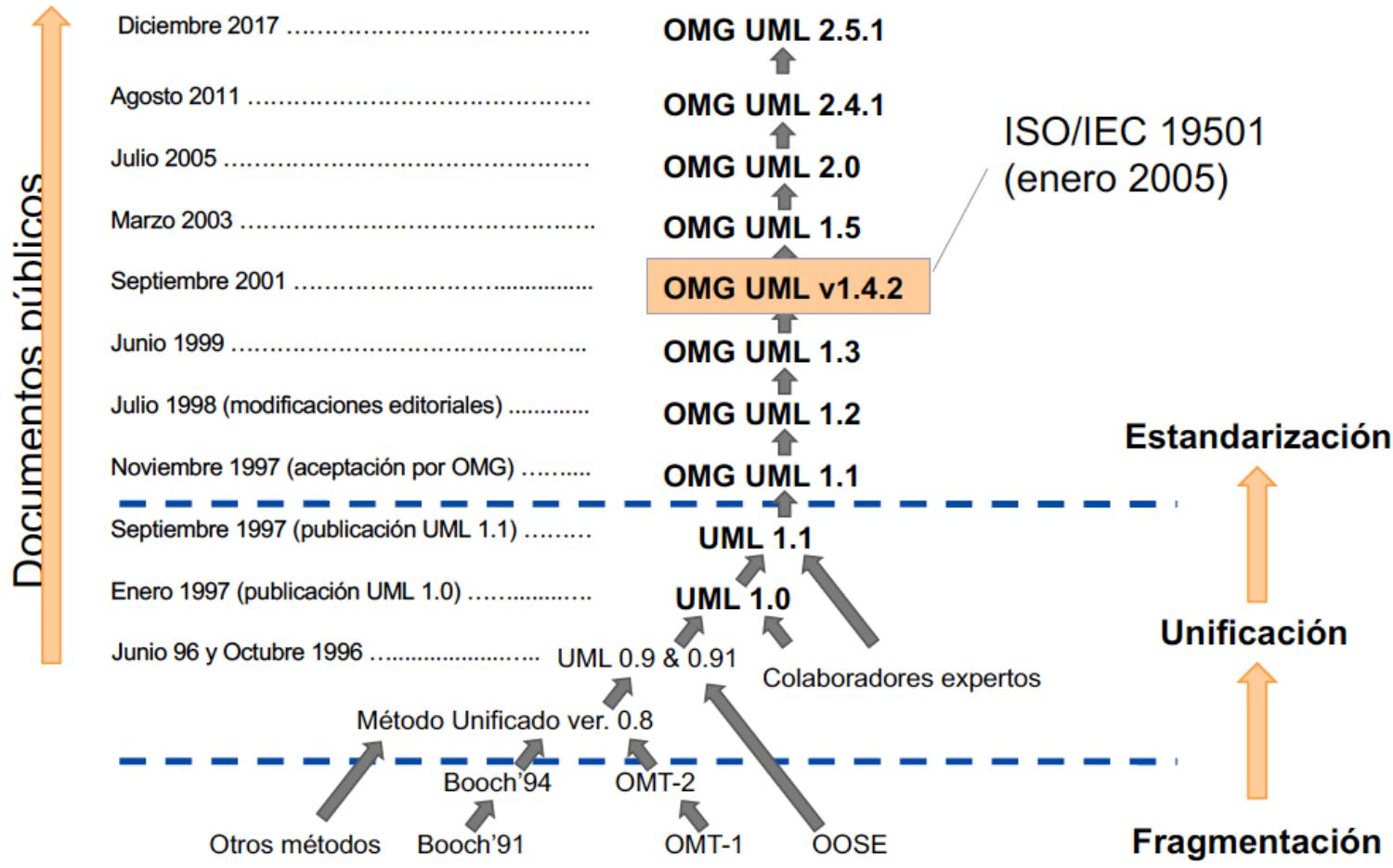
- 1994: Rumbaugh y Booch crean el Método Unificado
- 1995: Jacobson se une, publican Unified Method V0.8
- El método se reorienta hacia la creación de un lenguaje universal de modelado, naciendo UML
- 1996: Se forma un consorcio para desarrollar UML 1.0
- 1997: Se estandariza UML 1.0 por el OMG
- 1998-1999: Se publican versiones UML 1.2 y 1.3 con mejoras editoriales y semánticas
- 2001-2005: UML 1.4.2 se convierte en estándar ISO
- 2005: Se libera UML 2.0, la última versión estable es la 2.5.1 (2017)



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

### Evolución



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

### Ventajas de la Unificación

- Reunir los puntos fuertes de cada método
- Idear nuevas mejoras
- Proporcionar estabilidad al mercado
  - Proyectos basados en un lenguaje maduro
  - Aparición de potentes herramientas
- Eliminar confusión en los usuarios

### Objetivos en el diseño de UML

- Modelar sistemas desde los requisitos hasta los artefactos ejecutables utilizando técnicas OO
- Abordar la complejidad de sistemas grandes y críticos
- Ser un lenguaje útil tanto para personas como para máquinas
- Encontrar equilibrio entre expresividad y simplicidad



# Introducción al modelado de software y UML

## Técnicas orientadas a objeto: Lenguaje Unificado de Modelado

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos (modelos) de un sistema que involucra una gran cantidad de software, desde una perspectiva OO

- UML es una notación, no es un proceso
- Se han definido muchos procesos para UML
- Rational ideó *RUP*, “*Proceso Unificado de Rational*”
- Utilizable para sistemas que no sean software



# Índice

1. Introducción al modelado de software y UML
  1. El rol de los modelos en la ingeniería del software
  2. Técnicas orientadas a objeto: Lenguaje Unificado de Modelado
  3. El modelo conceptual de UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# Introducción al modelado de software y UML

## El modelo conceptual de UML

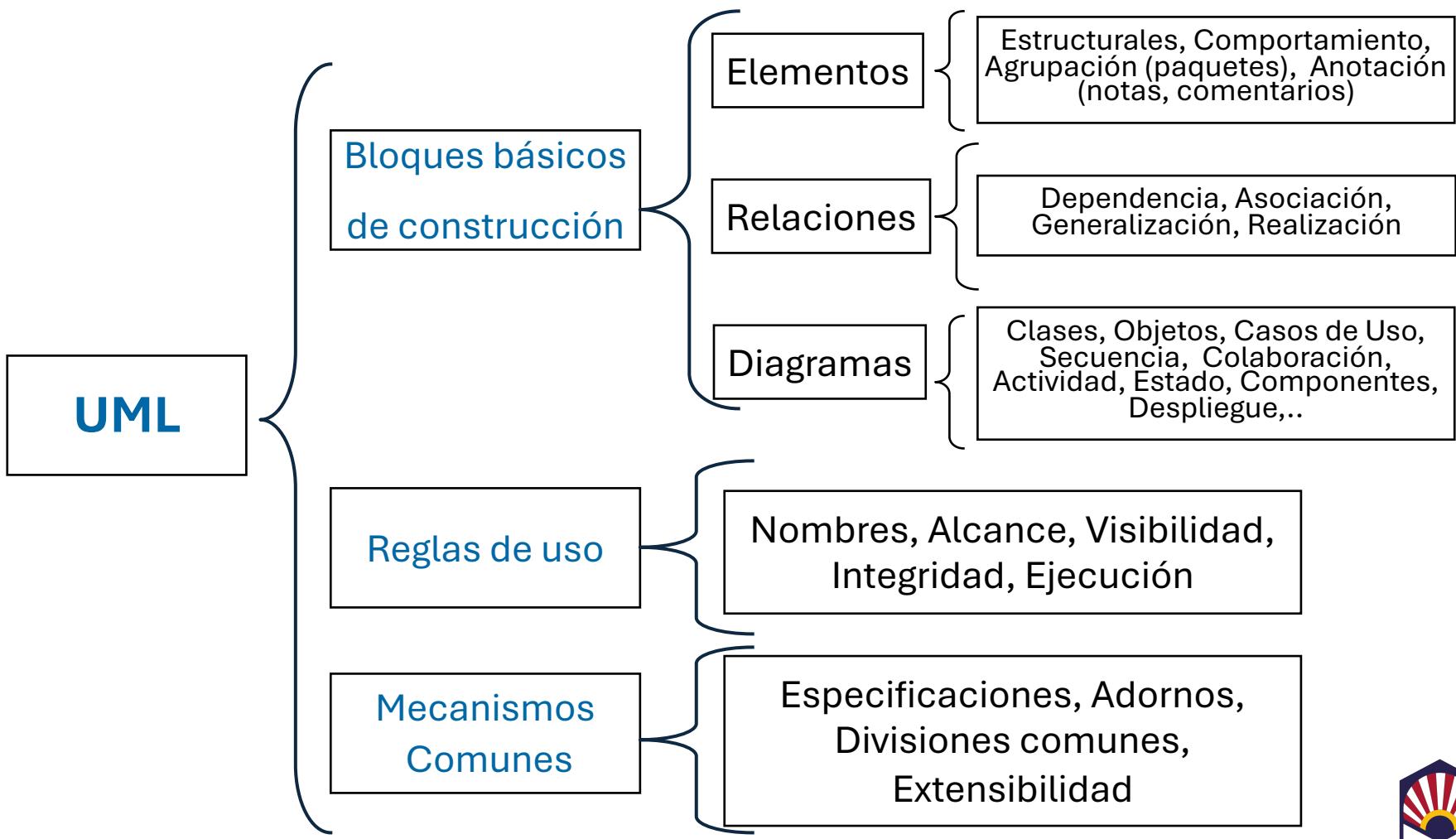
Elementos clave para comprender UML:

- **Bloques básicos:**
  - Elementos
  - Relaciones
  - Diagramas
- **Reglas de combinación:** Dictan cómo se pueden ensamblar los bloques
- **Mecanismos comunes:** Aplicables en todo lenguaje UML



# Introducción al modelado de software y UML

## El modelo conceptual de UML



# Introducción al modelado de software y UML

## El modelo conceptual de UML

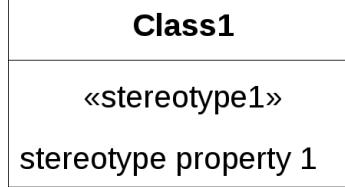
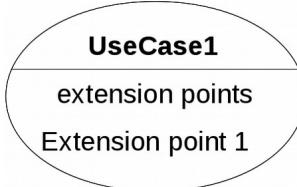
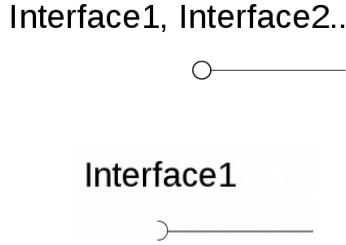
- Bloque de construcción en UML
  - Elementos
    - Bloques básicos de la programación orientada a objetos
    - Son “abstracciones” de primera clase dentro de un modelo
  - Relaciones
    - Conectan los diferentes elementos entre sí
  - Diagramas
    - Representación gráfica de un conjunto de elementos y sus relaciones entre sí
- Hay 4 tipos de elementos en UML
  - Elementos estructurales
  - Elementos de comportamiento
  - Elementos de agrupación
  - Elementos de anotación



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Principales elementos estructurales de UML

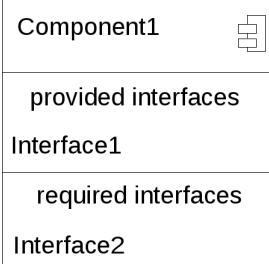
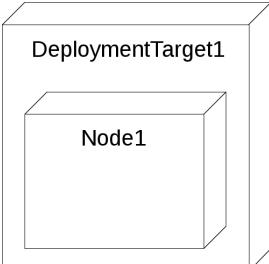
Elemento	Notación	Descripción
Clase		Es un contenedor (clasificador) que especifica una clasificación de objetos, incluyendo las características sobre su estructura y comportamiento
Caso de uso		Es una descripción de un conjunto de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular
Interfaz		Es una declaración de características públicas y obligaciones que constituyen un servicio concreto. Especifican un “contrato” con obligaciones en forma de pre o postcondiciones, orden, interacciones...



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Principales elementos estructurales de UML

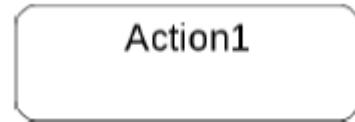
Elemento	Notación	Descripción
Componente		Unidad modular con interfaces definidas (expuestas en puertos) que puede ser reemplazado en su entorno. Su interior queda oculto, solo accesible por medio de las interfaces proveídas. Puede indicar dependencias (interfaces requeridas)
Artefacto		Elemento de información que se utiliza o produce mediante un proceso de desarrollo de software (ficheros, modelos, scripts, tablas de la base de datos, entregables...)
Nodo		Representan dispositivos hardware o entornos software de ejecución. Es un recurso computacional sobre el cual se despliegan los artefactos para su ejecución. Puede estar conformado por otros nodos internamente



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Principales elementos de comportamiento de UML

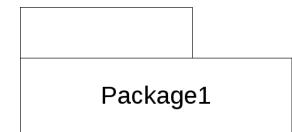
Elemento	Notación	Descripción
Mensaje		Medio de comunicación para el intercambio de información o la expresión de interacciones entre elementos. Pueden ser de diferentes tipos (respuesta, destrucción) y tener diferentes propiedades (asíncrono o no, de solo lectura), que modifican ligeramente su notación
Estado		Modela una situación en la ejecución durante la cual se cumple alguna condición (explícita o implícita) asociada a su nombre
Acción		Unidad fundamental de especificación de comportamiento en UML. Usualmente tiene un conjunto de entradas y produce un conjunto de salidas. Modifican el estado del sistema sobre el que se ejecutan



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- **Elementos de agrupación de UML**
  - Organizan y estructuran los modelos UML
  - El principal elemento de agrupación es el **paquete**, que organiza elementos estructurales, de comportamiento e incluso otros paquetes
  - Los paquetes actúan como “espacio de nombres”, definiendo el alcance de sus elementos
  - Pueden establecerse relaciones entre ellos (import, merge...)
- **Elementos de anotación de UML**
  - Añaden explicaciones y aclaraciones en los modelos UML
  - El principal elemento de notación es el **comentario**
  - Los comentarios describen y aclaran los elementos, pero no afectan la semántica de los modelos



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- Relaciones en UML
  - Permiten modelar los enlaces entre diferentes elementos estructurales
  - Proporcionan información adicional:
    - Multiplicidad: Indica el número de instancias de una clase que pueden estar asociadas con otras. Tipos:
      - 1 (no más de uno)
      - 0..1 (cero a uno)
      - 0..\* (cero a muchos)
      - 1..\* (uno a muchos)
      - \* (muchos)
    - Nombres de roles: Identifican los extremos de una asociación
  - UML maneja cuatro clases de relaciones: dependencia, asociación, generalización y realización



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- Relación de Dependencia:
  - Relación semántica donde un cambio de un elemento (independiente) puede afectar a otro (dependiente)



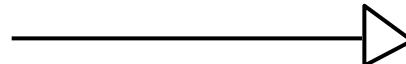
- Relación de Asociación:
  - Describe un conjunto de enlaces que representan conexiones entre objetos
  - **Agregación** y **composición** son formas especiales de asociación que indican una relación estructural entre un conjunto y sus partes



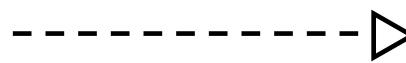
# Introducción al modelado de software y UML

## El modelo conceptual de UML

- Relación de Generalización:
  - Representa una relación de especialización/generalización entre elementos
  - Los objetos de un elemento especializado (hijos) comparten la estructura y comportamiento de un elemento generalizable (el padre)

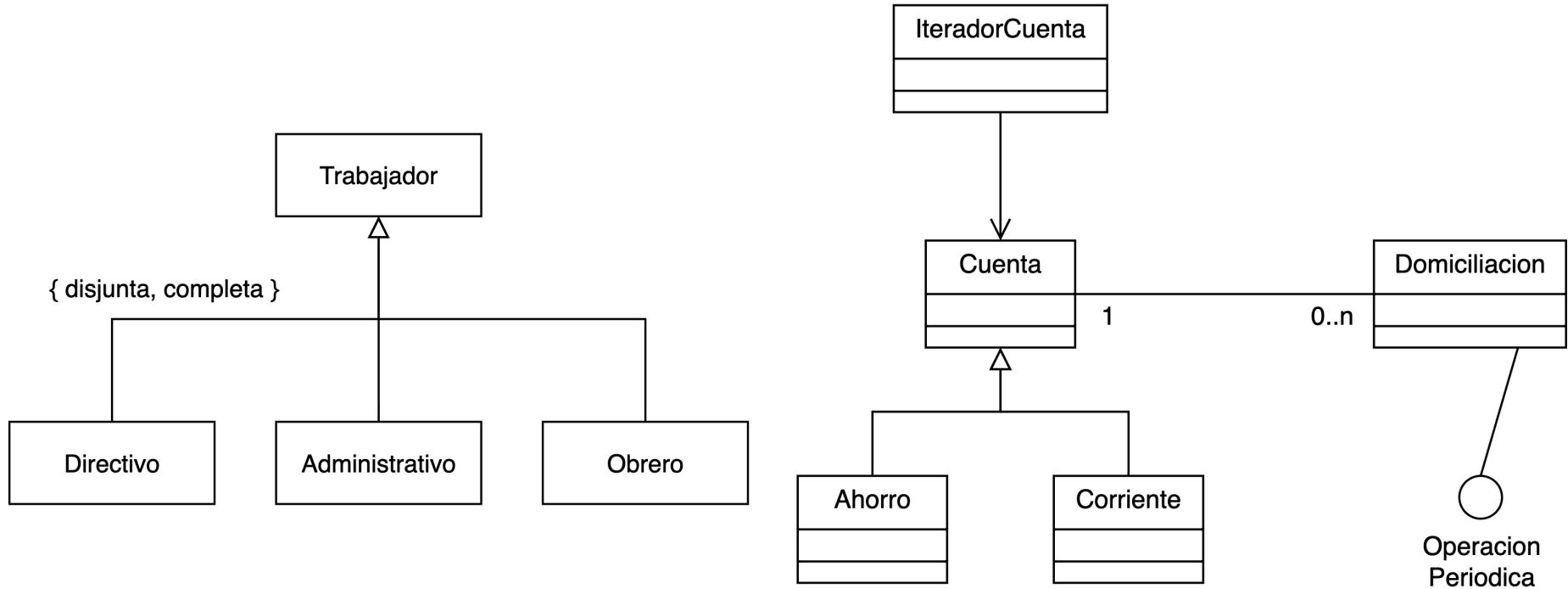


- Relación de Realización:
  - Es una relación semántica entre clasificadores
  - Un clasificador especifica un contrato que otro clasificador se compromete a cumplir
  - Las realizaciones pueden encontrarse en dos contextos:
    - Entre interfaces y las clases o componentes que las implementan
    - Entre casos de uso y las colaboraciones que los ejecutan



# Introducción al modelado de software y UML

## El modelo conceptual de UML



# Introducción al modelado de software y UML

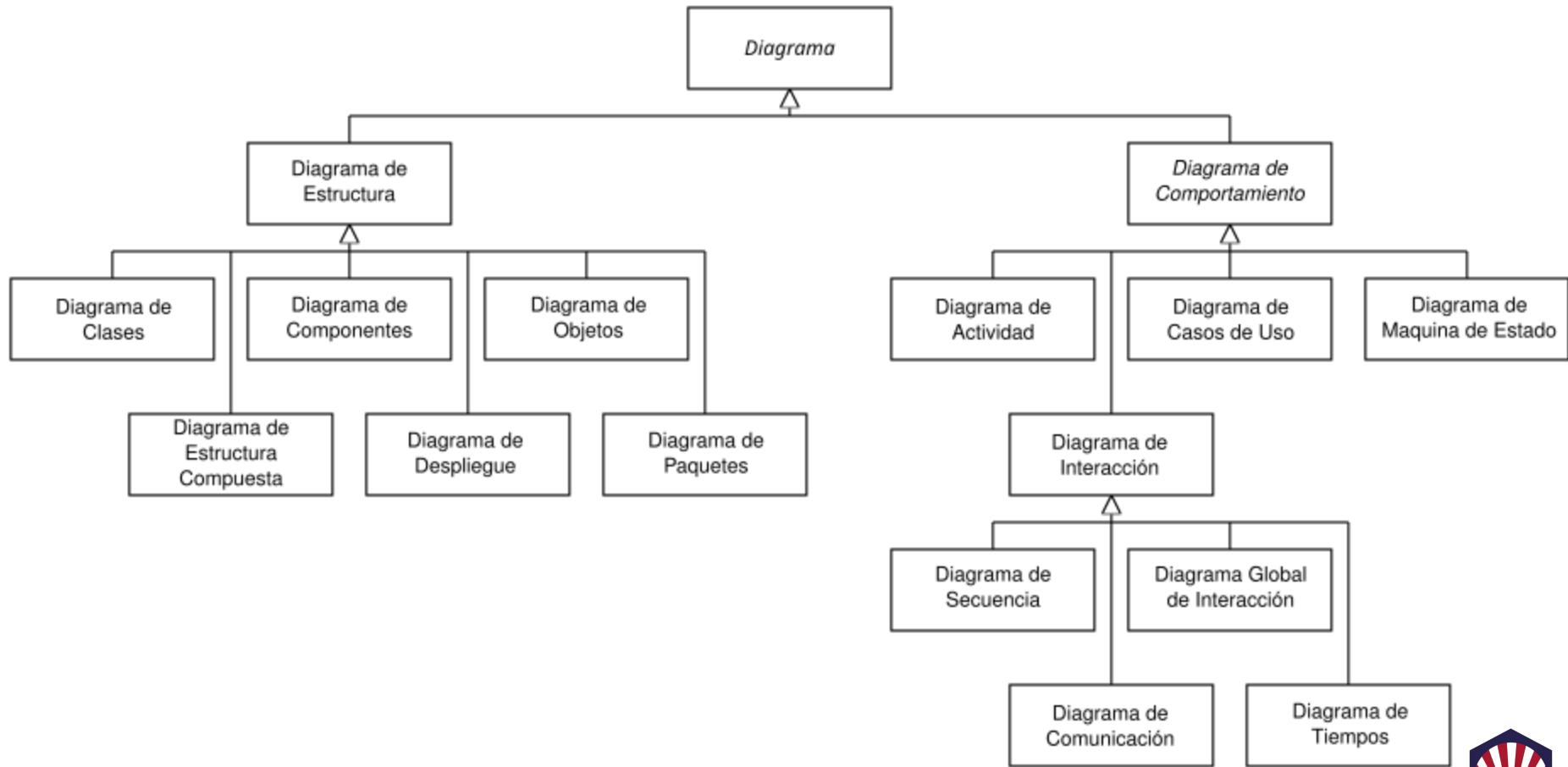
## El modelo conceptual de UML

- Un **diagrama** es la representación gráfica de un conjunto de elementos conectados
- Estos diagramas forman grafos conectados donde los vértices representan elementos y los arcos relaciones
- Sirven para visualizar un sistema desde diferentes perspectivas
- Un mismo elemento puede aparecer en varios diagramas, en algunos o en ninguno
  - **Diagramas Estructurales**
    - Visualizan, especifican, construyen y documentan los **aspectos estáticos** de un sistema
  - **Diagramas de Comportamiento**
    - Visualizan, especifican, construyen y documentan los **aspectos dinámicos** de un sistema



# Introducción al modelado de software y UML

## El modelo conceptual de UML



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- **Reglas en UML:**
  - Los bloques de construcción de UML no se pueden combinar de manera específica
  - UML tiene un número de reglas semánticas que especifican a qué debe parecerse un modelo bien formado:
    - **Nombres:** Especifican cómo llamar a los elementos, relaciones y diagramas
    - **Alcance:** Define el contexto que da un significado específico a un nombre
    - **Visibilidad:** Indica cómo se pueden ver y utilizar esos nombres por otros
    - **Integridad:** Asegura que los elementos se relacionen de manera apropiada y consistente entre sí
    - **Ejecución:** Establece lo que significa ejecutar o simular un modelo



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- **Mecanismos comunes en UML**
  - UML incorpora ciertos mecanismos comunes que pueden aplicarse en diferentes modelos para mejorar su interpretación y claridad
  - Estos mecanismos son:
    1. Especificaciones
    2. Adornos
    3. Divisiones comunes
    4. Extensibilidad



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- **Especificaciones**
  - Proporcionan una base semántica que incluye a todos los modelos de un sistema
  - Aseguran que cada elemento esté relacionado con otros de manera consistente
  - Se utilizan para enunciar los detalles del sistema
- **Adornos**
  - Notaciones gráficas claras para elementos UML
  - Se pueden agregar detalles adicionales que aclaren o complementen la información
  - Estos detalles se denominan **adornos**



# Introducción al modelado de software y UML

## El modelo conceptual de UML

- **Divisiones comunes**
  - Modelado desde la **generalidad** (abstracción) y/o lo **particular** (concreto)
  - Casi todos los bloques de construcción presentan esta posibilidad
    - Clase / Objeto
    - Casos de Uso / Instancias Casos de Uso
    - Componentes / Instancias de Componentes
- **Extensibilidad**
  - UML ofrece un lenguaje estándar para crear “planos de software”, pero puede no cubrir todos los matices en todos los dominios
  - Permiten adaptar y extender UML para las necesidades de un proyecto
  - Mecanismos de Extensibilidad:
    - Estereotipos: Crean nuevos tipos de bloques de construcción derivados de los existentes
    - Valores etiquetados: Añaden nueva información a las propiedades de los elementos
    - Restricciones: Extienden la semántica de un bloque de construcción



# Introducción al modelado de software y UML

## El modelo conceptual de UML

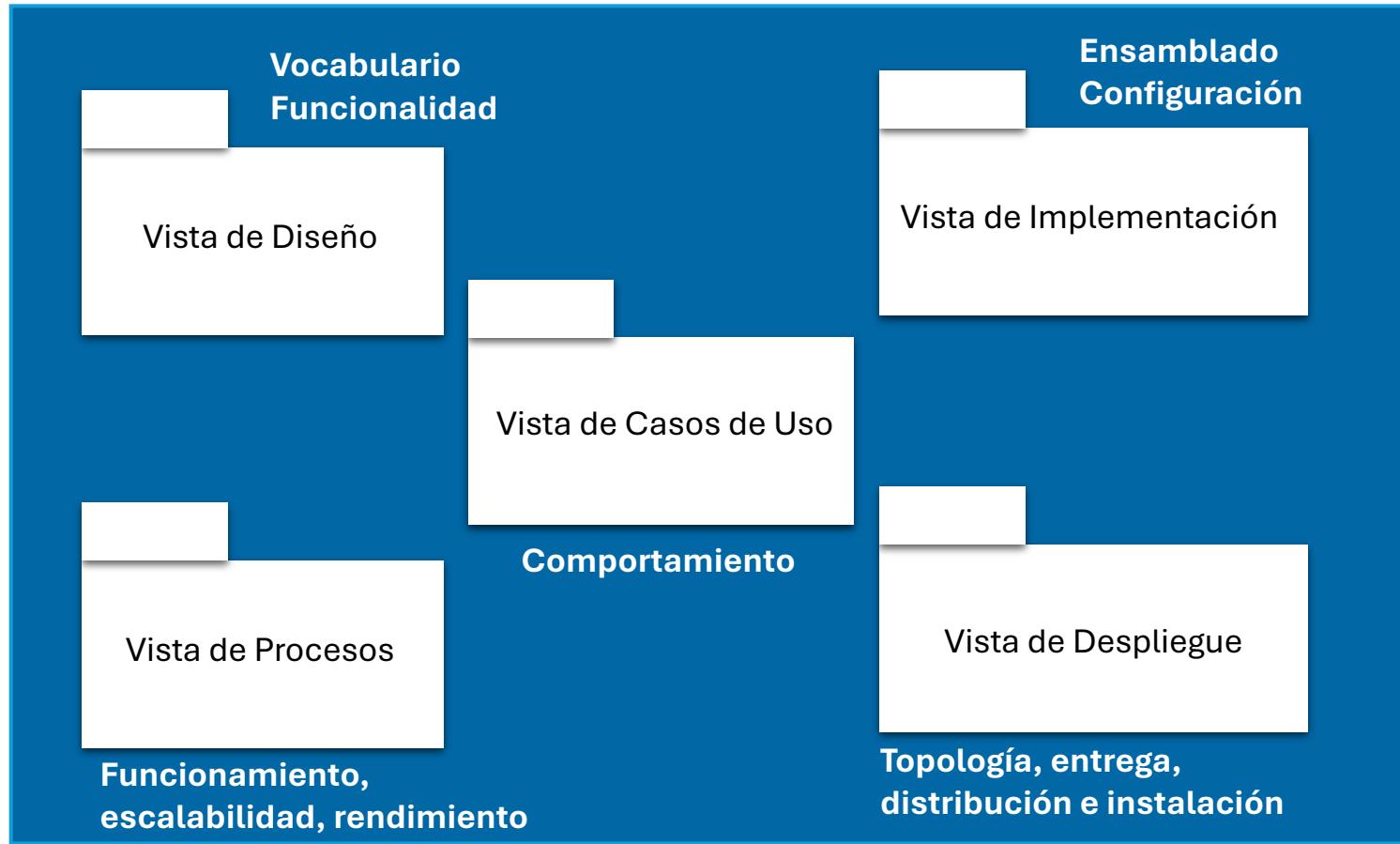
- La **visualización, especificación, construcción y documentación** de sistemas complejos requieren múltiples perspectivas
- Diferentes personas implicadas en el desarrollo tienen agendas variadas y miran el sistema de forma distinta a lo largo del ciclo de vida.
- En UML, la arquitectura de un sistema se describe mejor mediante cinco vistas interrelacionadas, cada una centrada en un aspecto particular del sistema.



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Vistas Interrelacionadas



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Vistas Interrelacionadas

- **Vista de Casos de Uso**
  - Describe el comportamiento del sistema desde la perspectiva de los usuarios finales, analistas y encargados de las pruebas
  - Utiliza:
    - **Casos de uso** para aspectos estáticos
    - **Diagramas de Interacción, Estados y Actividades** para aspectos dinámicos
- **Vista de Diseño**
  - Incluye clases, interfaces y colaboraciones que forman el vocabulario del problema y su solución
  - Soporta principalmente los requisitos funcionales del sistema
  - Utiliza:
    - **Diagramas de Clases y Objetos** para aspectos estáticos
    - **Diagramas de Interacción, Estados y Actividades** para aspectos dinámicos



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Vistas Interrelacionadas

- **Vista de Procesos**
  - Comprende los hilos y procesos que gestionan sincronización y concurrencia
  - Considera crecimiento, rendimiento y funcionamiento del sistema.
  - Utiliza los mismos diagramas que la vista de diseño, enfocándose en **clases activas** que representan hilos y procesos
- **Vista de Implementación**
  - Describe componentes y archivos que se utilizan para ensamblar y poner en marcha el sistema físico
  - Se centra en la gestión de versiones de componentes independientes que pueden combinarse de diferentes maneras
  - Utiliza:
    - **Diagramas de Paquetes** para aspectos estáticos
    - **Diagramas de Interacción, Estados y Actividades** para aspectos dinámicos



# Introducción al modelado de software y UML

## El modelo conceptual de UML

### Vistas Interrelacionadas

- **Vista de Despliegue**
  - Muestra los nodos que forman la topología hardware del sistema
  - Se ocupa de la distribución, entrega e instalación de las partes del sistema físico
  - Utiliza:
    - **Diagramas de Despliegue** para aspectos estáticos
    - **Diagramas de Interacción, Estados y Actividades** para aspectos dinámicos

Cada una de estas vistas puede existir de forma independiente y también interactuar entre sí



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
  4. Diagrama de Interacción
    - Diagrama de Secuencia
    - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- Un **caso de uso** describe cómo un sistema lleva a cabo una serie de acciones que generan un resultado valioso para un actor
- Se utilizan para capturar el comportamiento deseado del sistema, sin especificar su implementación
- Facilita la comprensión del sistema a los desarrolladores, los expertos en el dominio y los usuarios finales
- Ayudar a validar la arquitectura y verificar el sistema durante su evolución
- Deben mostrar solo los comportamientos esenciales, sin ser demasiado genéricos ni específicos



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- Los **casos de uso** se pueden utilizar como base para establecer casos de prueba durante el desarrollo del sistema, pueden ser:
  - Aplicados a subsistemas: excelente de pruebas de regresión
  - Aplicados al sistema completo: ideales para pruebas del sistema e integración
- Pasos para construir el modelo de casos de uso:
  1. Establecer el límite del sistema
  2. Definir a los actores
  3. Encontrar los casos de uso
    - a. Especificar el caso de uso
    - b. Indicar los flujos alternativos clave
  4. Repetir hasta que sistema, actores y casos de uso sean estables



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Componentes del modelo

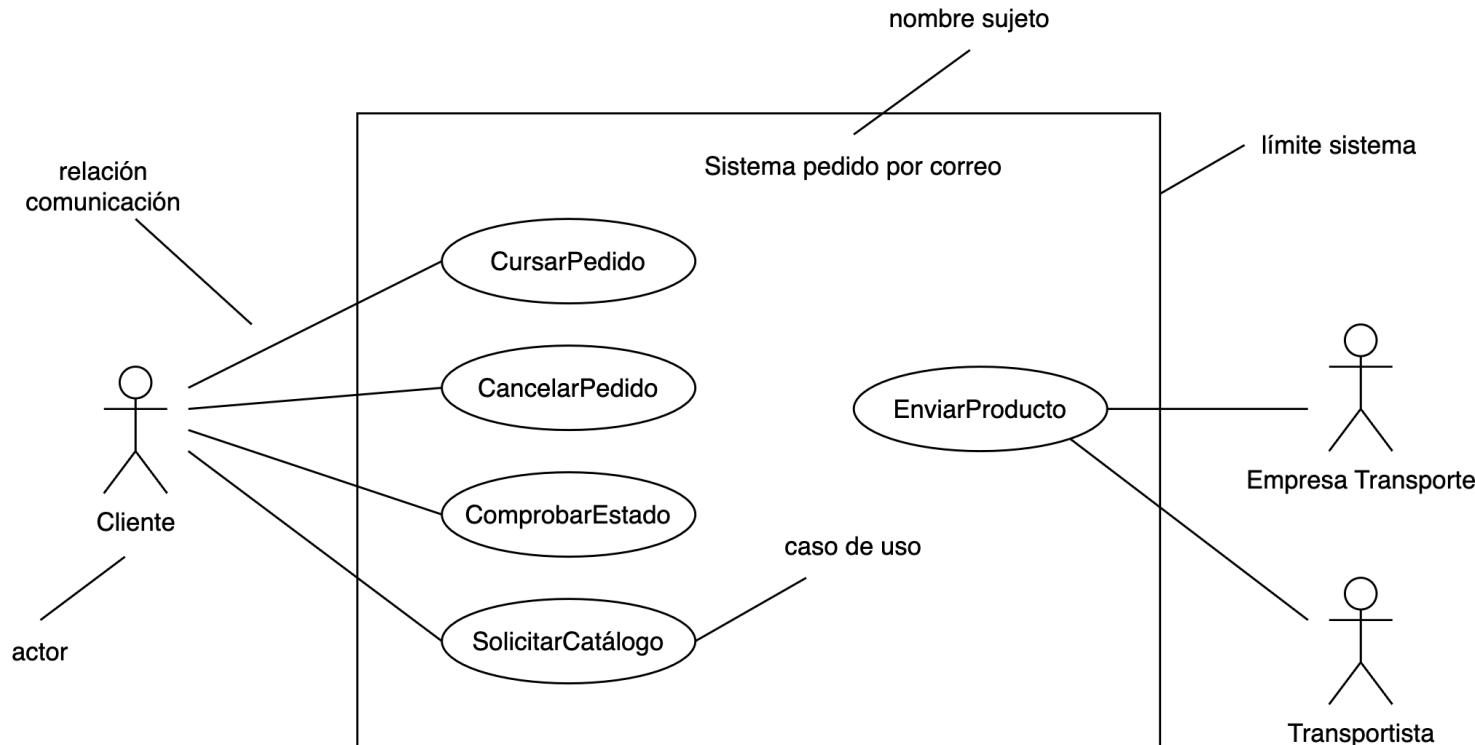
- **Límite del sistema:** Delimita el alcance del sistema que se modela (representado como un rectángulo)
- **Actores:** Roles de personas o sistemas que interactúan con el sistema
  - La misma persona puede actuar como diferentes actores
  - Actores **principales** activan casos de uso; actores **secundarios** interactúan
  - El nombre del actor describe su papel
  - Representan como “muñeco de palo”
  - Pueden especializarse mediante relaciones de generalización
- **Casos de uso:** Representan lo que los actores pueden hacer con el sistema (representados como elipses)
- **Relaciones:** Conectan actores y casos de uso, indicando la comunicación entre ellos



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- Un **diagrama de casos de uso** es un diagrama que muestra un conjunto de casos de uso, actores y sus relaciones
- Al igual que los demás diagramas pueden contener notas y restricciones



Fuente: J. Arlow, I. Neustadt. "UML 2". Anaya. 2005



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Especificación de los casos de uso

- El **comportamiento** de un caso de uso se describe mediante un flujo de eventos claro y comprensible para personas ajenas al sistema
- En este flujo de eventos se debe incluir:
  - Inicio y fin del caso de uso
  - Interacción con actores
  - Objetos intercambiados
- Además, se deberá incluir el **flujo principal** y los **flujos alternativos** del comportamiento
- Los casos de uso se determinan observando y precisando las secuencias de interacción con los actores desde el punto de vista del usuario



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Especificación de los casos de uso

- Aunque no existe un formato estandarizado, es habitual describir la especificación en formato tabular con los siguientes campos:
  - **Nombre** del caso de uso
  - **ID** del caso de uso
  - **Objetivo:** breve descripción
  - Contexto o **precondiciones**
  - **Actores** implicados: principal y, si es necesario, secundarios
  - **Escenario principal**
  - **Extensiones:** describen flujos alternativos
  - **Postcondiciones**



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Especificación de los casos de uso

Caso de uso	Cancelar pedido
Identificador	CU3
Objetivo	El cliente desea cancelar un pedido realizado anteriormente que aún no ha recibido
Contexto	El pedido debe existir en el sistema y no haberse tramitado aún
Actor principal	Cliente
Flujo principal	<ol style="list-style-type: none"><li>1. El sistema solicita al cliente el código del pedido</li><li>2. El cliente introduce el código del pedido a cancelar</li><li>3. El sistema localiza los datos del pedido</li><li>4. El cliente confirma que quiere eliminar el pedido</li><li>5. El sistema elimina los artículos del pedido</li><li>6. El sistema devuelve el dinero del pedido</li></ol>
Flujos alternativos	<p>3a. El pedido no existe. Se termina el caso de uso.</p> <p>3b. El pedido se encuentra en estado enviado. Se termina el caso de uso sin cancelar el pedido.</p>



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Especificación de los casos de uso

- Analizar los pasos del caso de uso para identificar claramente el flujo principal y los posibles flujos alternativos
- Flujo principal: captura el escenario “perfecto” sin errores, desviaciones o interrupciones
- Desviaciones del flujo principal:
  - Simples: Ramificaciones en el flujo principal
  - Complejas: Escenarios alternativos
- Flujos alternativos: se enfocan en errores y excepciones, sin volver al flujo principal
- Limitación de flujos alternativos:
  - Seleccionar los más importantes
  - Agrupar flujos similares



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- **Tipos de relaciones en UML:**

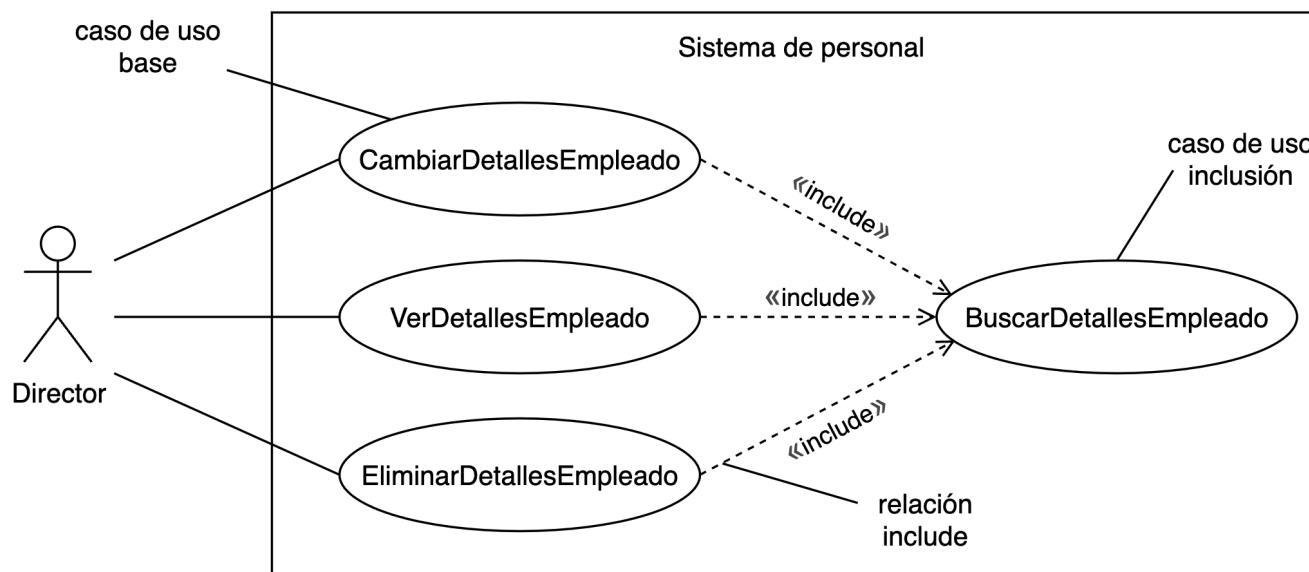
- **Comunicación:** Entre los actores y el sistema, determina entradas y salidas en el diagrama
- **Herencia:** Puede ocurrir entre actores o entre casos de uso
  - Entre actores: Abstacta similitudes en su interacción con el sistema
  - Entre casos de uso: Los casos de uso hijo pueden heredar, modificar o agregar características del caso de uso padre
  - En UML 2, los casos de uso no tienen atributos ni operaciones; la herencia afecta solo a relaciones, pre/postcondiciones y flujos
  - Recomendación: Usar la herencia solo si simplifica el modelo, ya que puede ser difícil de entender y mantener



# El Modelo de Comportamiento en UML

# Diagrama de Casos de Uso

- **Inclusión:** El caso de uso base incorpora el comportamiento de otro caso de uso en un punto específico del caso base
    - Evita repetir flujos de eventos, reutilizando un caso de uso aparte (concepto de reutilización)
    - Se representa con la etiqueta «include»
    - El caso de uso incluido siempre es parte de otro caso de uso, nunca está aislado



Fuente: J. Arlow, I. Neustadt. "UML 2". Anaya. 2005



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- **Inclusión:** Debe incluirse en la especificación de todos los casos de uso base

Caso de uso: CambiarDetallesEmpleado
ID: 1
Breve descripción: El Director cambia los detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno
Precondiciones: 1. El Director se conecta al sistema
Flujo principal: 1. include(BuscarDetallesEmpleado) 2. El sistema muestra los detalles del empleado. 3. El Director cambia los detalles del empleado. ...
Postcondiciones: 1. Se han cambiado los detalles del empleado.
Flujos alternativos: Ninguno

Caso de uso: VerDetallesEmpleado
ID: 2
Breve descripción: El Director ve los detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno
Precondiciones: 1. El Director se conecta al sistema
Flujo principal: 1. include(BuscarDetallesEmpleado) 2. El sistema muestra los detalles del empleado. ...
Postcondiciones: 1. El sistema ha mostrado los detalles del empleado.
Flujos alternativos: Ninguno

Caso de uso: EliminarDetallesEmpleado
ID: 3
Breve descripción: El Director elimina los detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno
Precondiciones: 1. El Director se conecta al sistema
Flujo principal: 1. include(BuscarDetallesEmpleado) 2. El sistema muestra los detalles del empleado. 3. El Director elimina los detalles del empleado. ...
Postcondiciones: 1. Los detalles del empleado se han eliminado.
Flujos alternativos: Ninguno

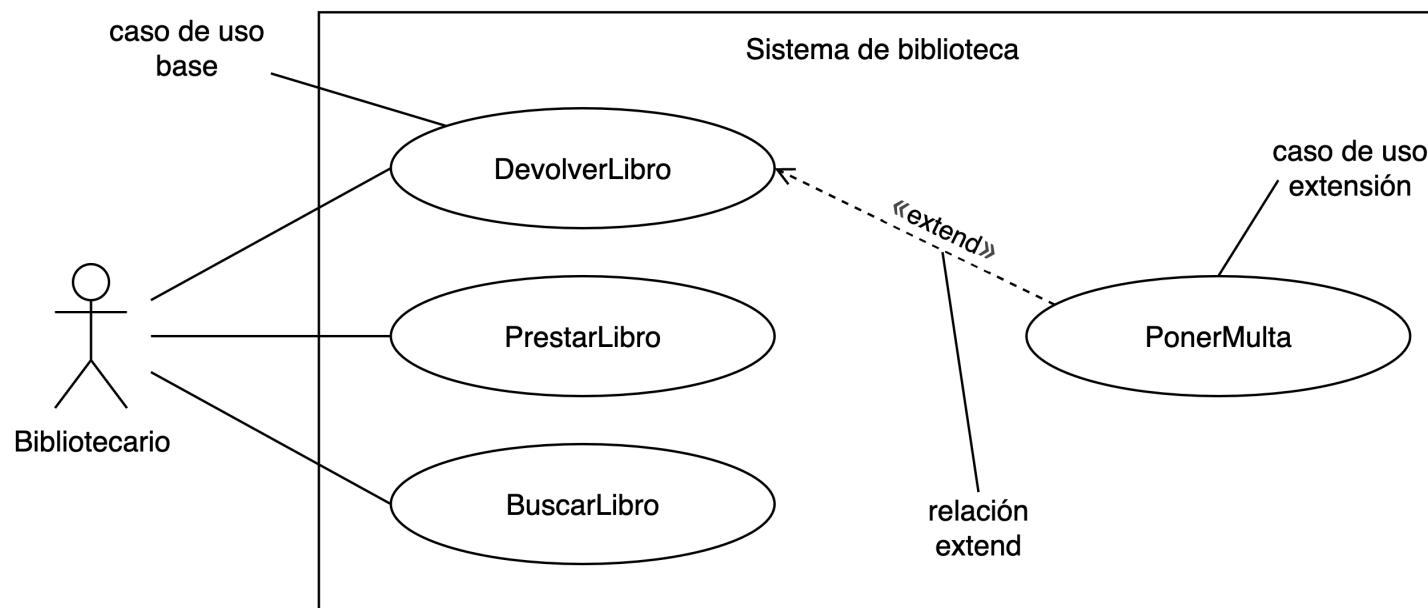
Fuente: J. Arlow, I. Neustadt. “UML 2”. Anaya. 2005



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- **Extensión:** El caso de uso origen añade comportamiento al caso de uso destino, pero el destino es válido sin la extensión (diferente de la inclusión)
  - Se usa para modelar comportamientos opcionales o subflujos que se activan bajo ciertas condiciones
  - Se representa con la etiqueta «extend»



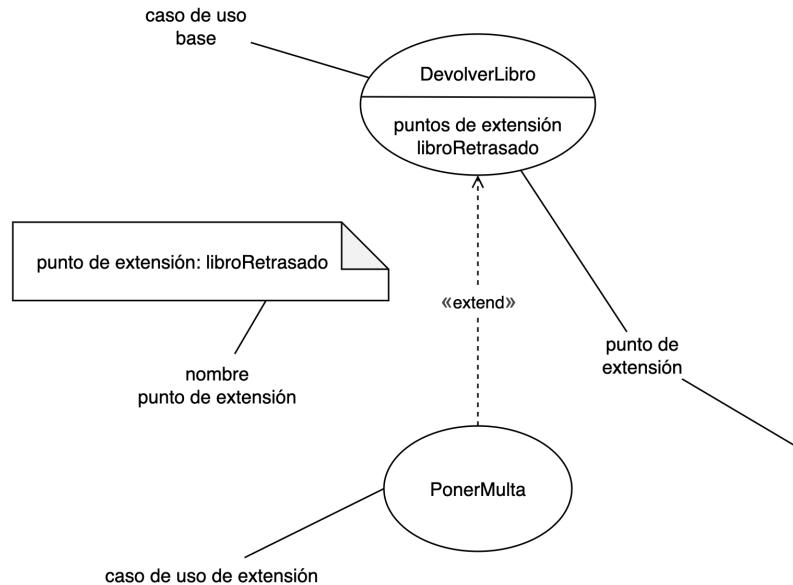
Fuente: J. Arlow, I. Neustadt. “UML 2”. Anaya. 2005



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

- **Extensión:** Debe mencionarse en la especificación de los casos de uso afectados
  - Indican puntos donde se puede extender del flujo principal, pero no forma parte de él
  - Útil para capturar casos excepcionales o extensiones no previstas



Caso de uso: DevolverLibro
ID: 9
Breve descripción: El Bibliotecario devuelve un libro prestado.
Actores principales: Bibliotecario
Actores secundarios: Ninguno
Precondiciones: 1. El Bibliotecario se conecta al sistema.
Flujo principal: 1. El Bibliotecario escribe el número del prestatario. 2. El sistema muestra los detalles del prestatario incluida la lista de libros prestados. 3. El Bibliotecario busca el libro a devolver en la lista de libros. punto de extensión: libroRetrasado 4. El Bibliotecario devuelve el libro. ...
Postcondiciones: 1. El libro se ha devuelto.
Flujos alternativos: Ninguno

Fuente: J. Arlow, I. Neustadt. “UML 2”. Anaya. 2005



# El Modelo de Comportamiento en UML

## Diagrama de Casos de Uso

### Trazabilidad

- El modelo de casos de uso está ligado a la especificación de requisitos, importante en la ingeniería de requisitos
- Es importante relacionar requisitos funcionales con casos de uso para garantizar:
  - Cada requisito funcional está cubierto por al menos un caso de uso
  - Cada caso de uso hace referencia a, al menos, un requisito funcional
- Algunas herramientas de modelado o de ingeniería de requisitos permiten vincular casos de uso y requisitos
- Alternativamente, se puede vincular a mano mediante una matriz de trazabilidad:
  - Detectan inconsistencias (falta de casos de uso o de requisitos)
  - Ayudan al seguimiento de requisitos durante el proyecto
  - Para utilizarlas es importante que tanto requisitos como casos de uso tengan ID



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
  4. Diagrama de Interacción
    - Diagrama de Secuencia
    - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

- Un **diagrama de actividades** modela un proceso como una secuencia de nodos conectados, similar a un diagrama de flujo (orientado a objetos)
- Usos principales:
  - Modelar visualmente el flujo de un caso de uso
  - Representar detalles de operaciones o algoritmos
  - Ofrece flexibilidad para diferentes tipos de flujos
- Elemento principal: la **actividad**:
  - Se asocia a un elemento (caso de uso, clase, interfaz, etc.) para explicar su comportamiento
  - Limita el contexto y puede hacer referencia a características (propiedades u operaciones) del elemento



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

- Las actividades constituyen una red de nodos conectados
- Tres tipos de nodos:
  - **Nodos de acción**: unidad atómica dentro de la actividad
  - **Nodos de control**: controlan el flujo
  - **Nodos de objeto**: representan a los objetos utilizados en la actividad
- Tipos de Flujos:
  - **Flujos de control**: conectan nodos para el flujo de la actividad
  - **Flujos de objeto**: conectan los objetos en la actividad
- Las actividades y sus nodos pueden tener precondiciones y postcondiciones
- Existen símbolos especiales para indicar los nodos de inicio y fin de la actividad



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

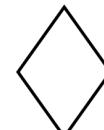
### Representación



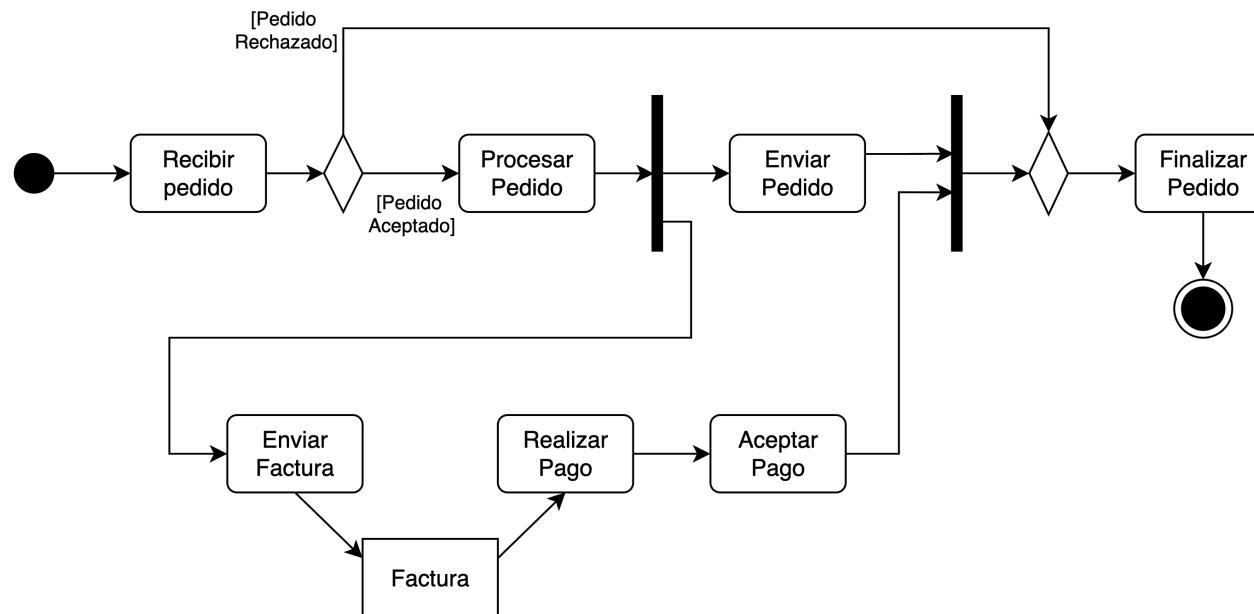
Nodo Acción



Nodo Objeto



Nodos de Control



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

- Las **transiciones** conectan un estado de actividad o acción con el siguiente
- Tipos de Transición:
  - **Secuenciales:** flujo simple entre actividades
  - **Bifurcaciones:** caminos alternativos definidos por expresiones booleanas
    - Una bifurcación tiene una transición de entrada y dos o más de salida, cada una con una condición booleana que se evalúa solo una vez
- Las condiciones en cada salida no deben solaparse y deben cubrir todas las opciones para asegurar un flujo continuo.

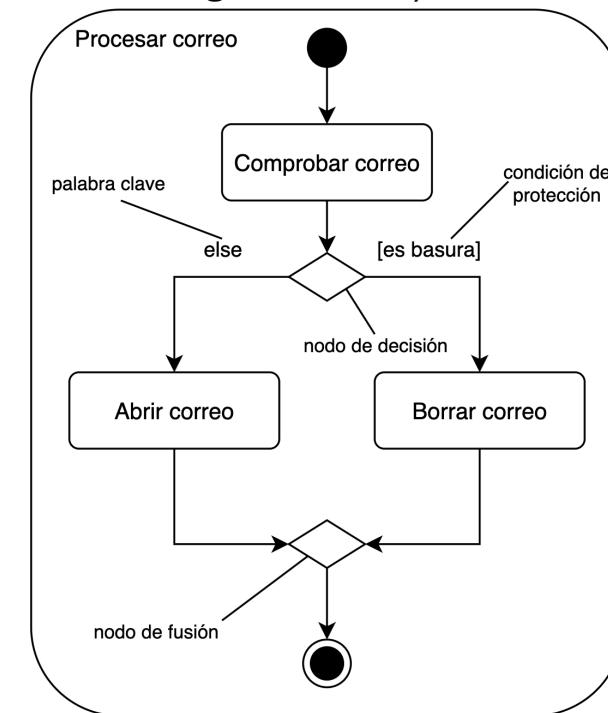


# El Modelo de Comportamiento en UML

## Diagrama de Actividades

### Nodos de control

- Permiten gestionar el flujo de control dentro de la actividad
  - Pueden haber múltiples nodos de inicio para representar concurrencia
  - El nodo de fin detiene todos los flujos (o solo uno, según el caso)
- Nodos de decisión
  - Un punto de entrada y varios puntos de salida
  - Cada salida tiene una condición exclusiva para definir el flujo
- Nodos de fusión
  - Fusionan varios flujos de entrada en uno solo de salida



# El Modelo de Comportamiento en UML

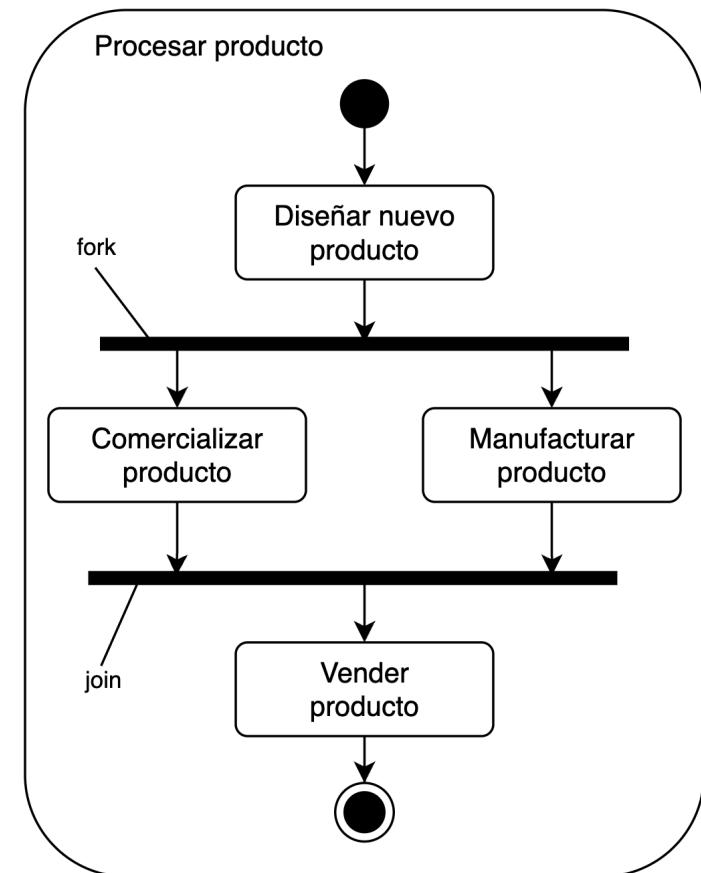
## Diagrama de Actividades

### Nodos de control: “fork” y “join”

- Se utilizan para crear subflujos concurrentes dentro de la actividad
  - El nodo “fork” divide un flujo en varios paralelos
  - El nodo “join” sincroniza flujos paralelos en uno solo

### Nodos de objeto

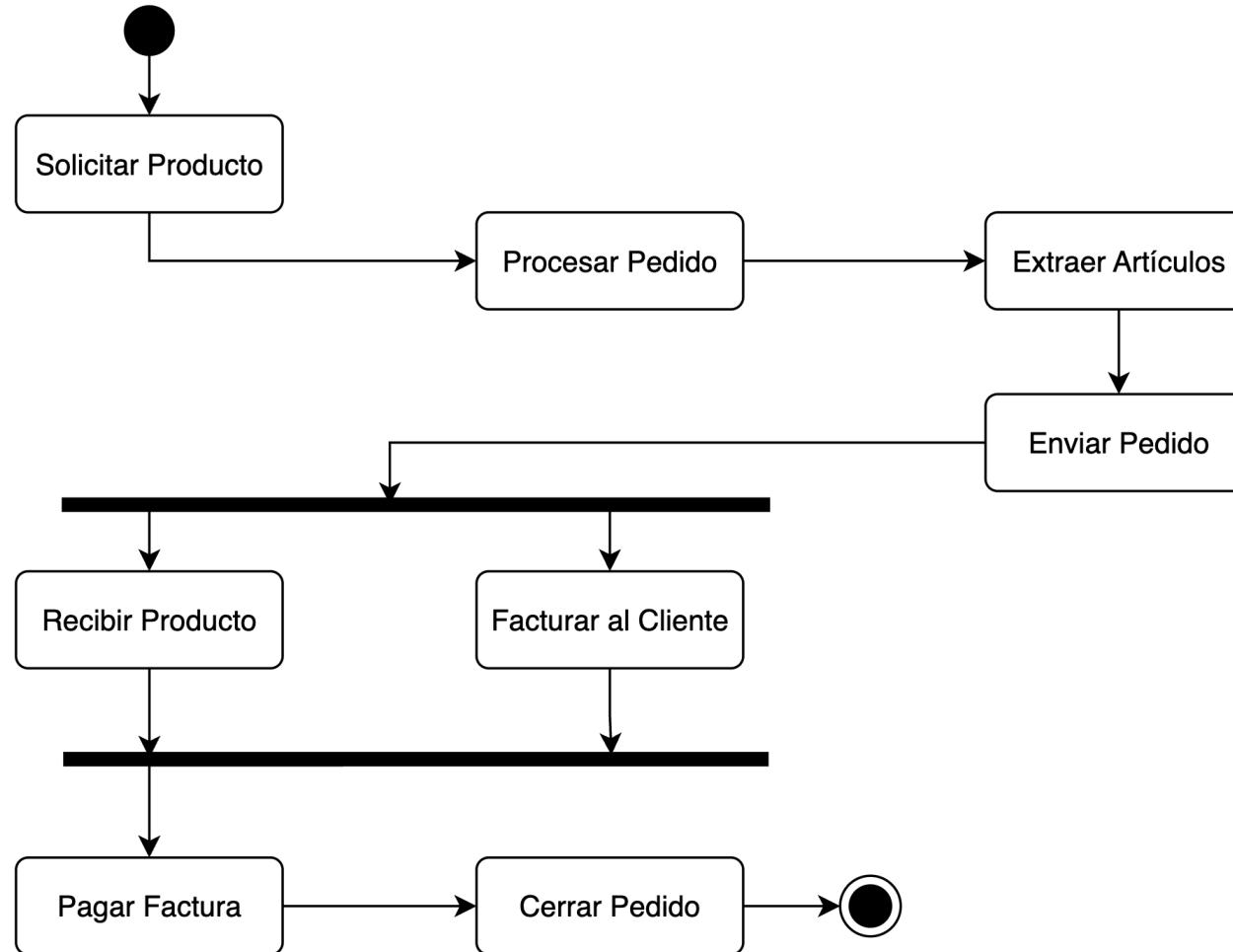
- Indican la disponibilidad de una instancia en un punto de la actividad
- Tienen flujos de entrada y salida de tipo objeto
- Representan la creación y uso de objetos por los nodos de acción



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

### División y unión

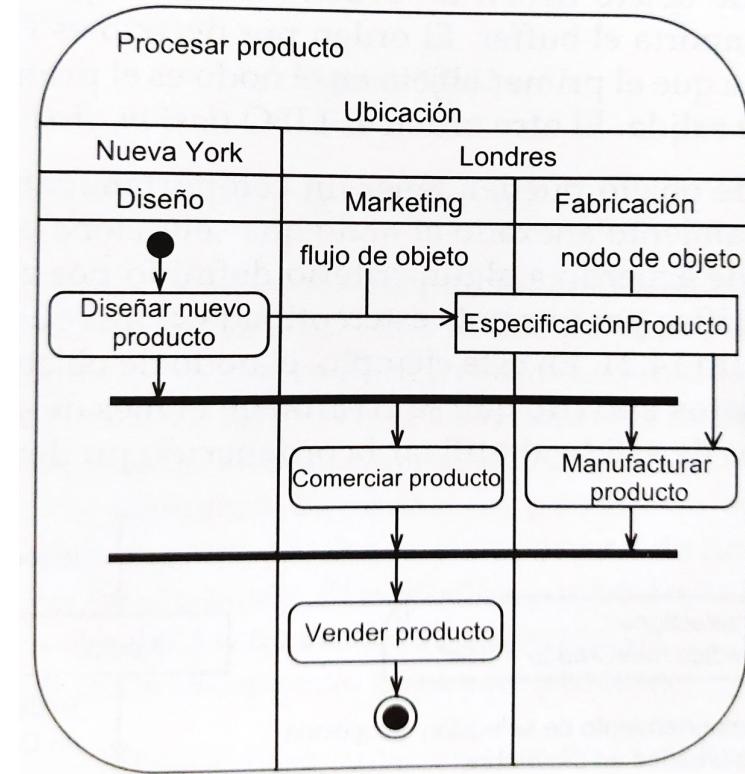


# El Modelo de Comportamiento en UML

## Diagrama de Actividades

### Particiones de Actividad

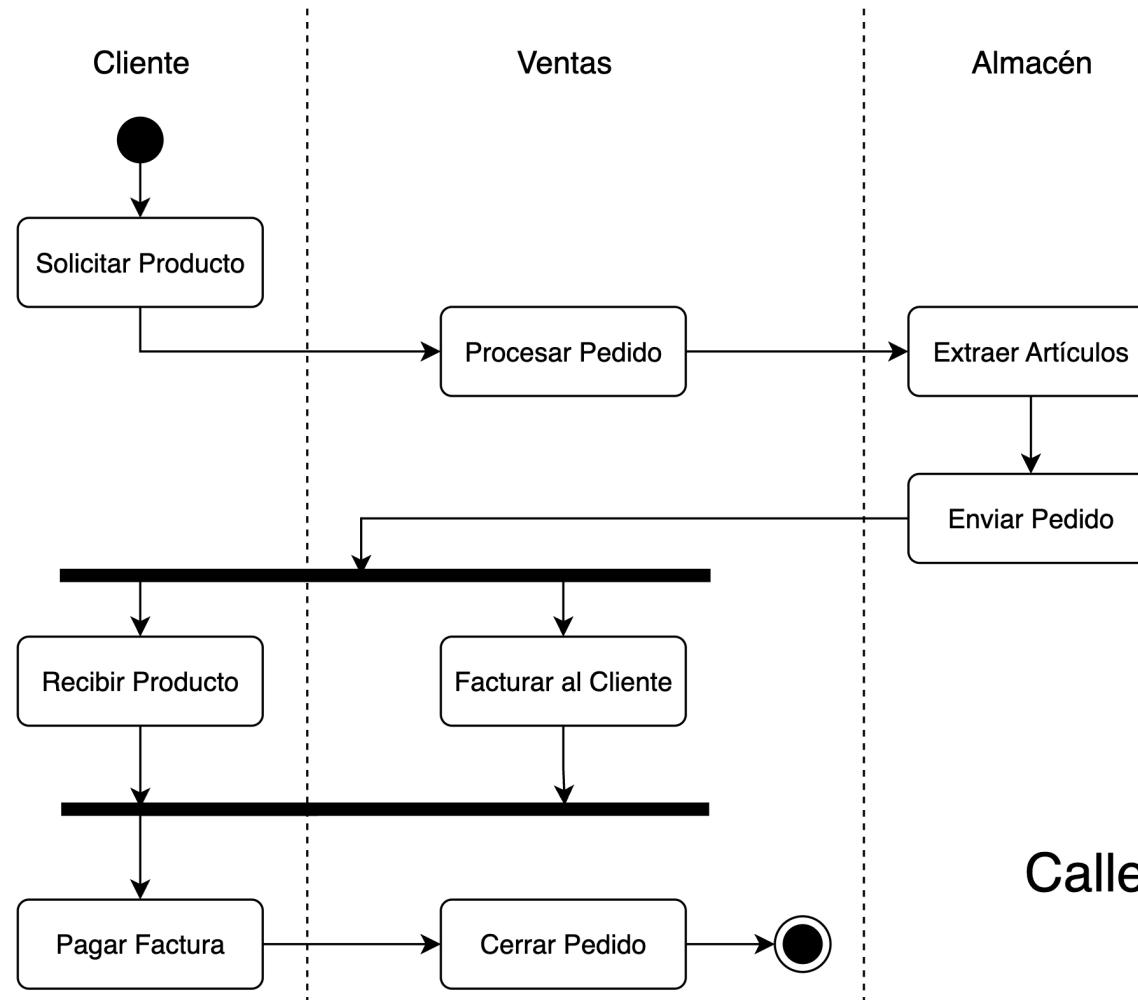
- Dividen el diagrama (horizontal o verticalmente) para mejorar su claridad
- También llamadas calles o carriles (*swimlanes*), con nombre único
- Agrupan acciones relacionadas
- Ayudan a asociar acciones a diferentes elementos
- Su interpretación es flexible según el modelador



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

### Calles o *Swimlanes*



Calles



# El Modelo de Comportamiento en UML

## Diagrama de Actividades

### Consejos y sugerencias

- Usar varios diagramas para capturar toda la dinámica del sistema; un solo diagrama no abarca todo
- Incluir solo los elementos esenciales para comprender el flujo
- Agregar solo los adornos necesarios para su claridad
- Nombrar el diagrama de forma que refleje su propósito
- Comenzar modelando el flujo principal; bifurcaciones, concurrencias y flujos de objetos pueden ir en diagramas secundarios si es necesario.



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
  4. Diagrama de Interacción
    - Diagrama de Secuencia
    - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

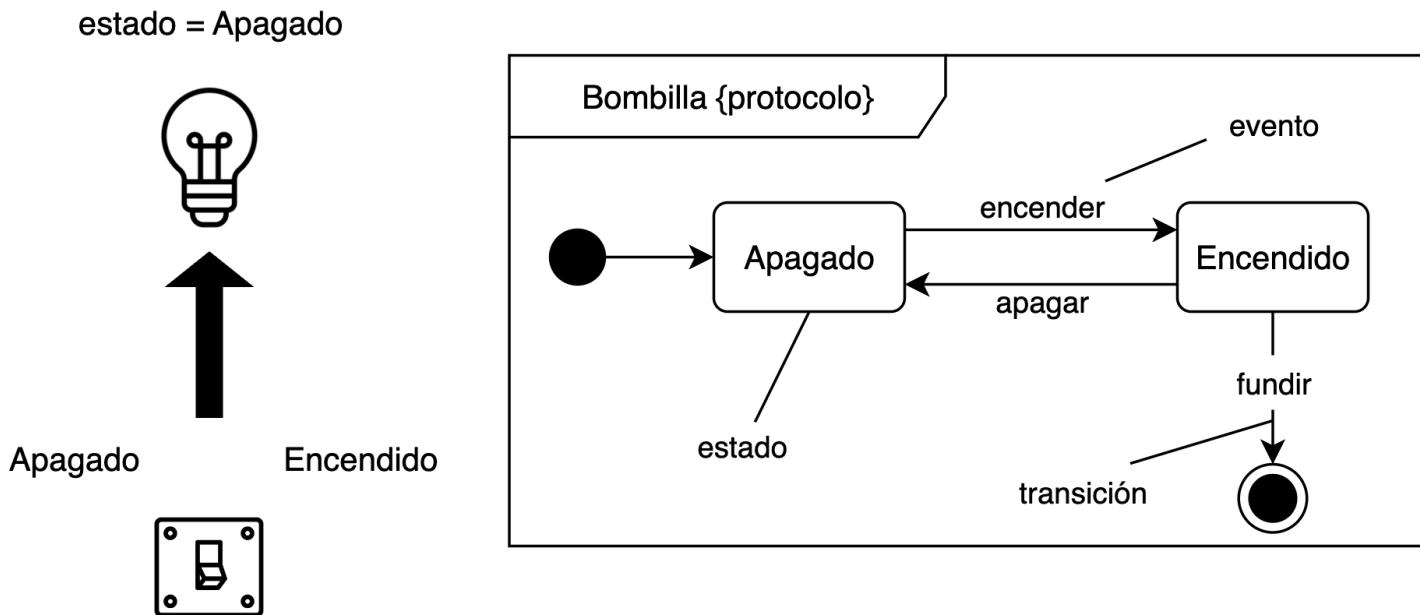
- Una máquina de estados define la secuencia de estados que un objeto atraviesa durante su vida, impulsada por eventos internos o externos
- Modela el comportamiento dinámico de elementos como clases, casos de uso o sistemas completos
- Relación con otros diagramas:
  - **Diagramas de interacción:** Modelan el comportamiento de varios objetos; la máquina de estados modela el comportamiento de solo uno
  - **Diagramas de actividades:** Se enfocan en el flujo de control entre actividades; no entre estados; el paso entre actividades ocurre al finalizar la actividad anterior



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

- Elementos de una máquina de estados:
  1. **Estados:** Condiciones en las que se encuentra un objeto
  2. **Transiciones:** Rutas posibles entre estados
  3. **Eventos:** Sucesos que ocurren en un momento determinado
- En el diagrama, los estados de inicio y fin se representan de forma diferente al resto de estados



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

### Estados

- Un estado es una condición o situación en la vida de un objeto en la que:
  - Satisface alguna condición
  - Realiza alguna actividad
  - Espera algún evento
- Un objeto permanece en un estado durante un tiempo finito
- Partes de un estado:
  - **Nombre:** Texto que identifica el estado, a veces puede ser anónimo
  - **Acciones de entrada/salida:** Acciones ejecutadas al entrar y salir del estado
  - **Transiciones internas:** Cambios manejados sin alterar el estado
  - **Actividades internas:** Operaciones que requieren tiempo y pueden interrumpirse por eventos

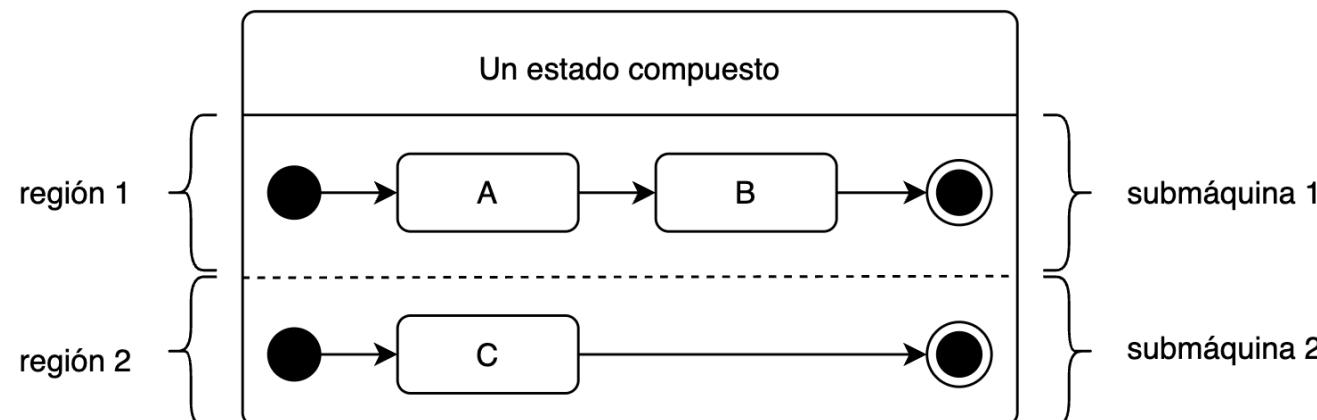


# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

### Estados Compuestos

- Son estados que contiene otros estados anidados en su interior (submáquinas)
- Los estados anidados heredan todas las transiciones de sus estados contenedores
- Pueden ser:
  - Sencillos: Con una sola región
  - Ortogonales: Con varias regiones
- Se pueden anidar a cualquier nivel, pero se recomienda no tener más de 2 o 3 niveles



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

### Transiciones

- Una transición es la relación entre dos estados que indica que un objeto realizará ciertas acciones y cambiará al segundo estado al ocurrir un evento y cumplir condiciones específicas
- Cuando ocurre este cambio, se dice que la transición se ha disparado
- Antes de dispararse, el objeto está en el estado origen; después, en el estado destino
- Se representa como una línea continua dirigida del origen al destino
- Una auto transición es el cambio donde el estado origen y destino es el mismo
- Puede tener múltiples orígenes (join, unión de estados concurrentes) o múltiples destinos (fork, división a varios estados concurrentes)



# El Modelo de Comportamiento en UML

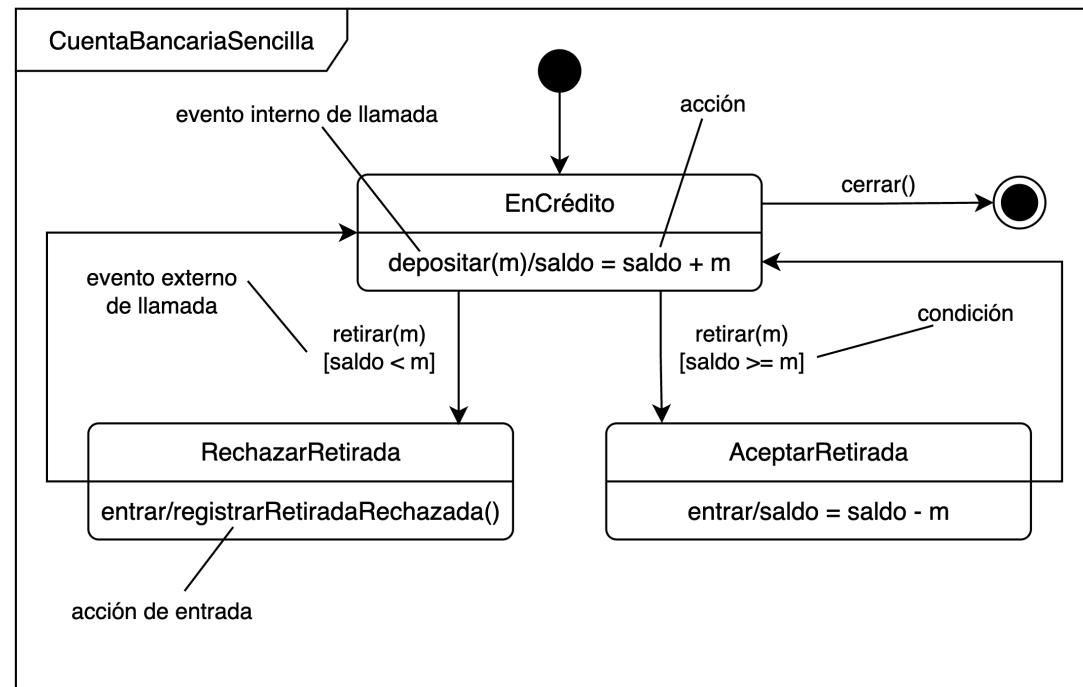
## Diagrama de Máquinas de Estado

### Transiciones

- Tienen tres elementos opcionales:

1. **Evento(s) de disparo:** Ocurrencias (internas o externas) que activan la transición
2. **Condición(es) de Guarda (o de protección):** Expresión booleana entre corchetes que se evalúa después de que ocurre el evento de disparo
3. **Acción(es)**

- Procesos a realizar cuando se activa la transición
- Pueden incluir llamadas a operaciones sobre el objeto, cambiar sus propiedades, etc.
- No puede ser interrumpida y se ejecuta hasta completarse



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

### Eventos

- Un evento es un acontecimiento significativo que ocurre en un momento y lugar específicos
- En las máquinas de estados, modelan los estímulos que producen un cambio de estado
- Los eventos pueden suceder externamente (representación sobre la transición) o internamente (dentro de un estado)
- Tipos de Eventos:
  1. Evento de Llamada
  2. Evento de Señal
  3. Evento de Tiempo
  4. Evento de Cambio



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

### Evento de Llamada

- Representan la solicitud de una operación específica en un objeto
- Al invocar una operación sobre otro objeto con una máquina de estados:
  - El control pasa del emisor al receptor
  - El evento activa la transición
  - La operación se completa
  - El receptor cambia a un nuevo estado y el control vuelve al emisor
- La operación debe estar en la lista de operaciones del objeto receptor, y debe respetar su firma (coherencia)

### Evento de Señal

- Consiste en el envío asíncrono de información, sin una operación asociada.
- Tiene una representación especial para el envío y recepción de señales.

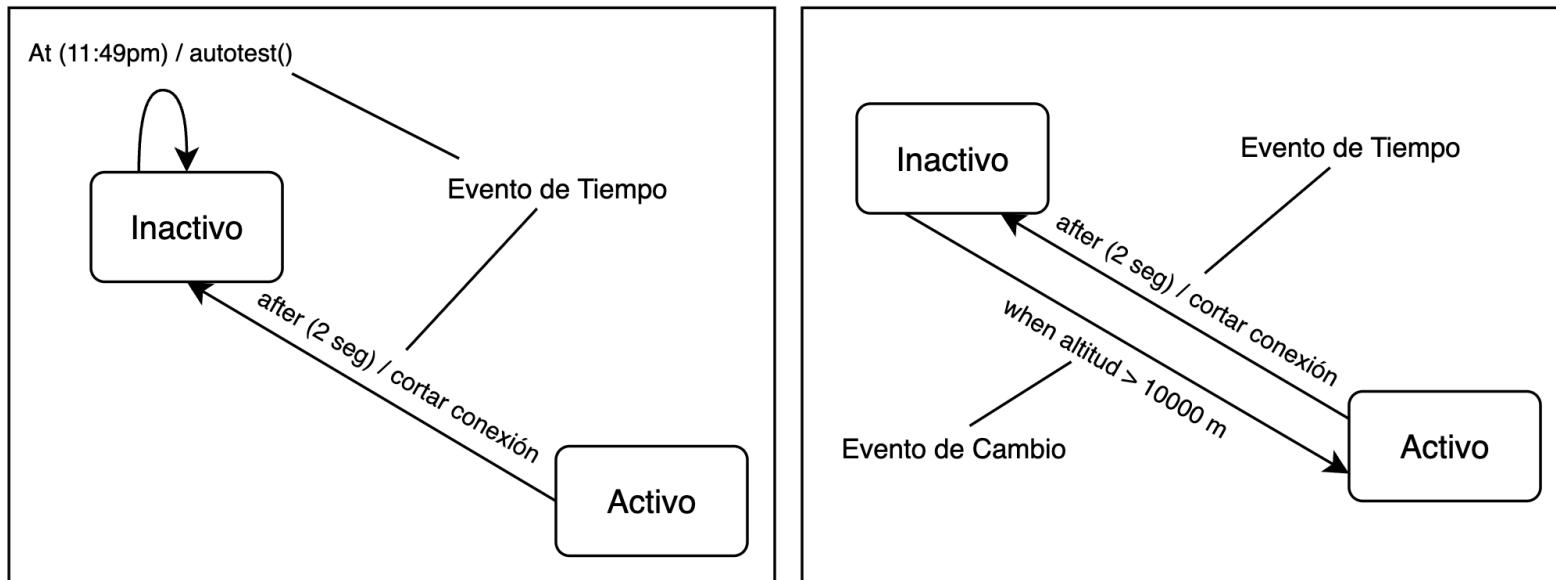


# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado

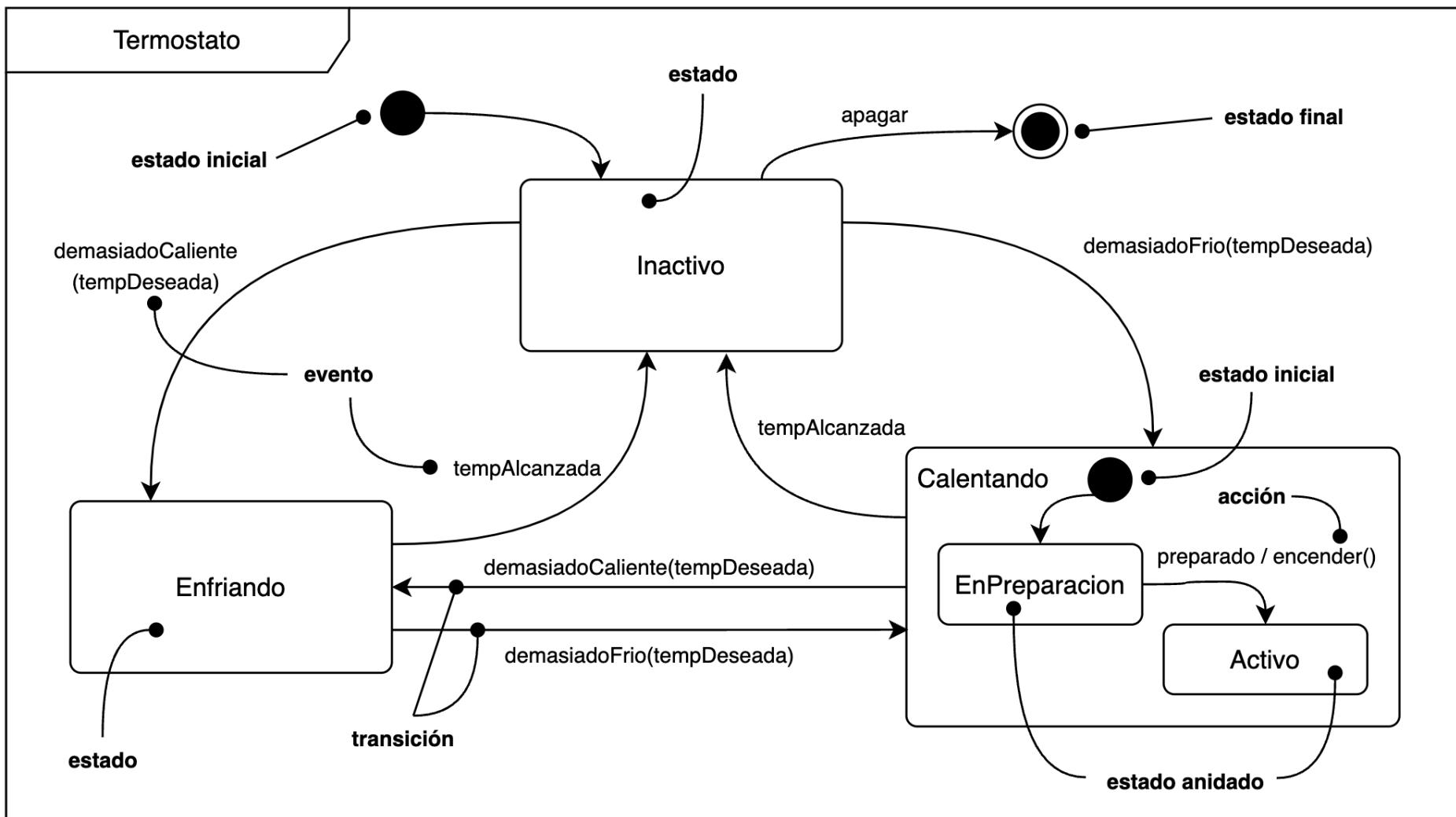
### Eventos de Tiempo y Cambio

- Un **evento de tiempo** representa el paso del tiempo, modelado con la palabra “*after*” seguido de una expresión. El tiempo se mide desde el momento en que se entra en el estado
- Un **evento de cambio** representa un cambio en el estado o el cumplimiento de una condición, modelado con la palabra “*when*” seguido de una expresión booleana



# El Modelo de Comportamiento en UML

## Diagrama de Máquinas de Estado



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
4. Diagrama de Interacción
  - Diagrama de Secuencia
  - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

- Modelan el **comportamiento dinámico** del sistema y el **flujo de control** en una operación
- Describe la **interacción** entre objetos a través de mensajes para cumplir ciertas tareas
- Las interacciones muestran un “comportamiento” y “realizan” (implementan) un caso de uso
- **Elementos principales:** objetos, enlaces y mensajes; también pueden incluir notas y restricciones



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

- Aunque existen cuatro tipos de diagramas de interacción, nos centraremos en:
  - **Diagrama de Secuencia:** Destaca el orden temporal de los eventos. Gráficamente es una tabla que representa a objetos en el eje X y mensajes en el eje Y en orden cronológico
  - **Diagrama de Colaboración:** Destaca la estructura de los objetos que envían y reciben los mensajes. Gráficamente es una colección de nodos y arcos
- Son semánticamente equivalentes
  - Se puede generar uno a partir del otro, sin perdida de información
  - Visualmente, sin embargo, esta información puede ser más difícil de percibir
- Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos en una interacción



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

Tipo	Ventajas	Desventajas
Secuencia	<ul style="list-style-type: none"><li>- Notación simple</li><li>- Ilustran mejor las secuencias</li></ul>	<ul style="list-style-type: none"><li>- Elaboración <i>rígida</i></li></ul>
Colaboración	<ul style="list-style-type: none"><li>- Elaboración <i>flexible</i></li><li>- Ilustran mejor condicionales, iterativas, concurrentes</li></ul>	<ul style="list-style-type: none"><li>- Notación compleja</li><li>- Ilustran peor las secuencias</li></ul>



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

### Interacciones y mensajes

- Una **interacción** es una unidad de comportamiento de un clasificador (por ejemplo, un caso de uso) que muestra cómo sucede una comunicación mediante el intercambio de mensajes
- Un **mensaje** representa un tipo específico de comunicación que puede invocar operaciones, crear o destruir instancias, o enviar señales
- Las interacciones modelan el comportamiento dinámico de las colaboraciones, incluyendo clases, interfaces, componentes, nodos y casos de uso
- Los aspectos dinámicos se muestran como flujos de control que pueden incluir desde secuencias simples hasta flujos más complejos con bifurcaciones, iteraciones, recursión y concurrencia



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

### Notación

Sale

una clase

:Sale

una instancia

s1:Sale

una instancia  
nombrada

*Sintaxis:* return := nombreMensaje(parametro : tipoParam,...) : tipoRetorno



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

### Tipos de mensajes

Nombre	Notación	Significado
Mensaje síncrono	→	El emisor espera hasta recibir respuesta del receptor
Mensaje asíncrono	→	El emisor prosigue su ejecución sin esperar respuesta del receptor
Retorno de mensaje	<-----	Devolución del foco de control
Creación de objeto	→ <<create>> :A	Creación de una instancia
Destrucción de objeto	→ *	Destrucción de una instancia
Mensaje encontrado	• →	Llegada de un mensaje sin especificar emisor
Mensaje perdido	→ •	Mensaje que no llega a su destino (error)



# El Modelo de Comportamiento en UML

## Diagrama de Interacción

### Secuenciación

- Cuando un objeto envía un mensaje a otro, este a su vez puede enviar un mensaje a un tercer objeto, desencadenando otros mensajes y formando una secuencia
- Una secuencia comienza en algún proceso o hilo y continuará mientras este exista
- Cada proceso o hilo define un flujo de control separado, ordenando mensajes cronológicamente
- Para visualizar mejor dicha secuencia, se utiliza un número de secuencia (1, 1.1, 1.2, etc.), que indica el orden de los mensajes desde el inicio



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
4. Diagrama de Interacción
  - Diagrama de Secuencia
  - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

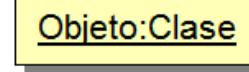
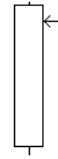
- Muestran la interacción entre componentes del sistema desde el punto de vista temporal
- La interacción se representa desde el punto de vista de paso de mensajes entre objetos o actores a lo largo del tiempo
- Utilidad:
  - Describir procesos internos entre diferentes módulos
  - Describir comunicaciones con otros sistemas o con actores
- Se suelen asociar a los casos de uso, mostrando como estos se realizan a través de interacciones entre sociedades de objetos



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

### Elementos

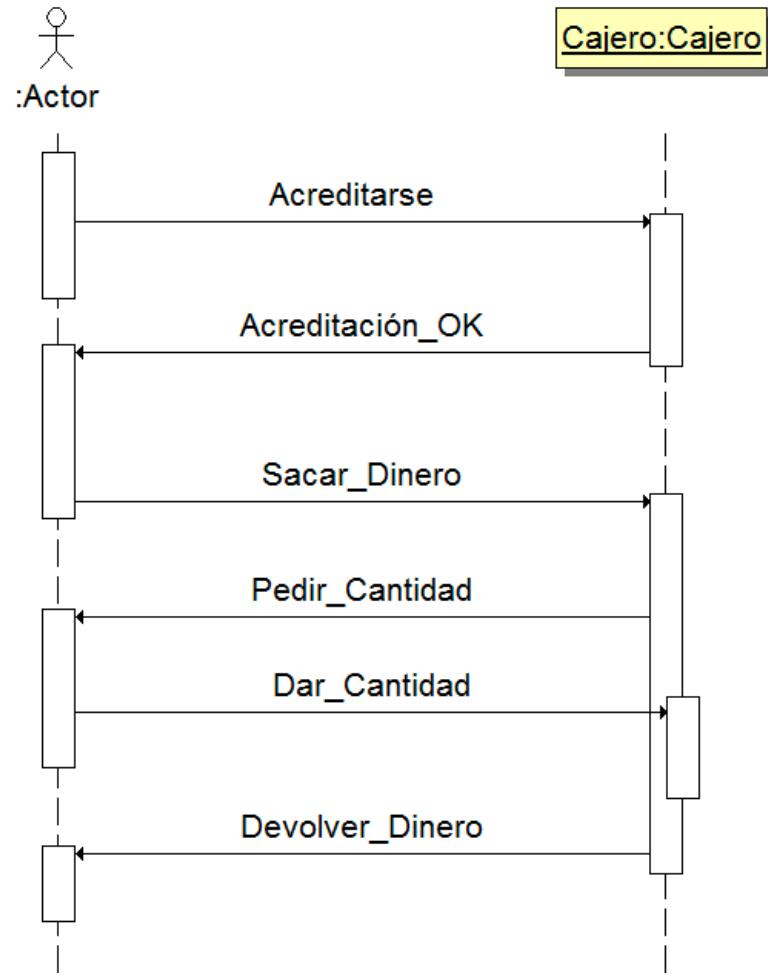
Nombre	Notación
Actor	 :Actor
Objeto	
Foco de control	
Actores/objetos desconocidos	
Fin de una “línea de vida”	X



# El Modelo de Comportamiento en UML

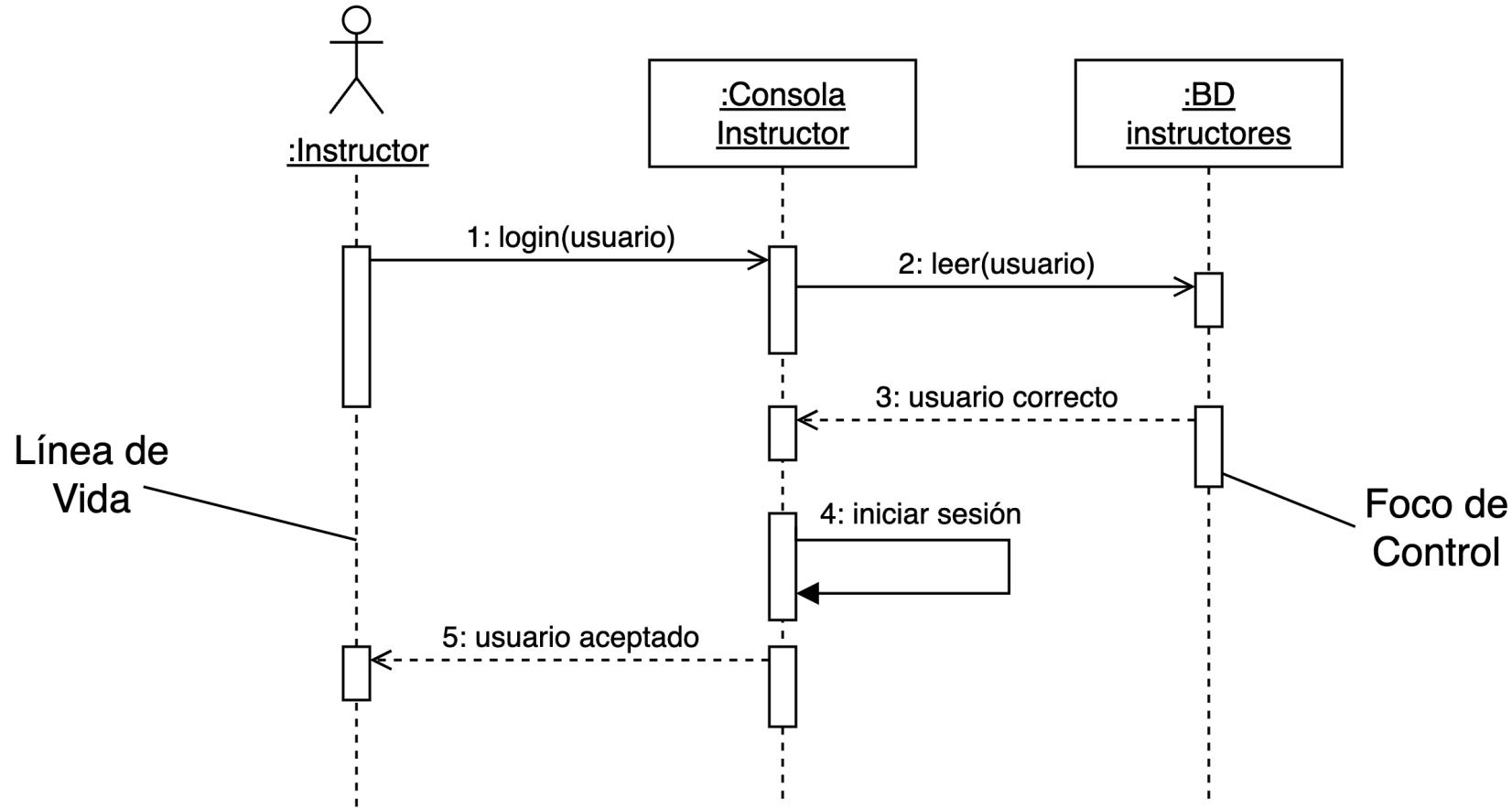
## Diagrama de Interacción – Diagrama de Secuencia

- Cada actor u objeto tiene un eje vertical (línea de vida) que muestra su existencia a lo largo del tiempo
- El paso de mensajes se indica con una línea horizontal entre los objetos, además de la descripción del mensaje
- Cuando el objeto/actor se encuentra activo se representa un rectángulo sobre la línea de tiempo, tan grande como tiempo se encuentre activo



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

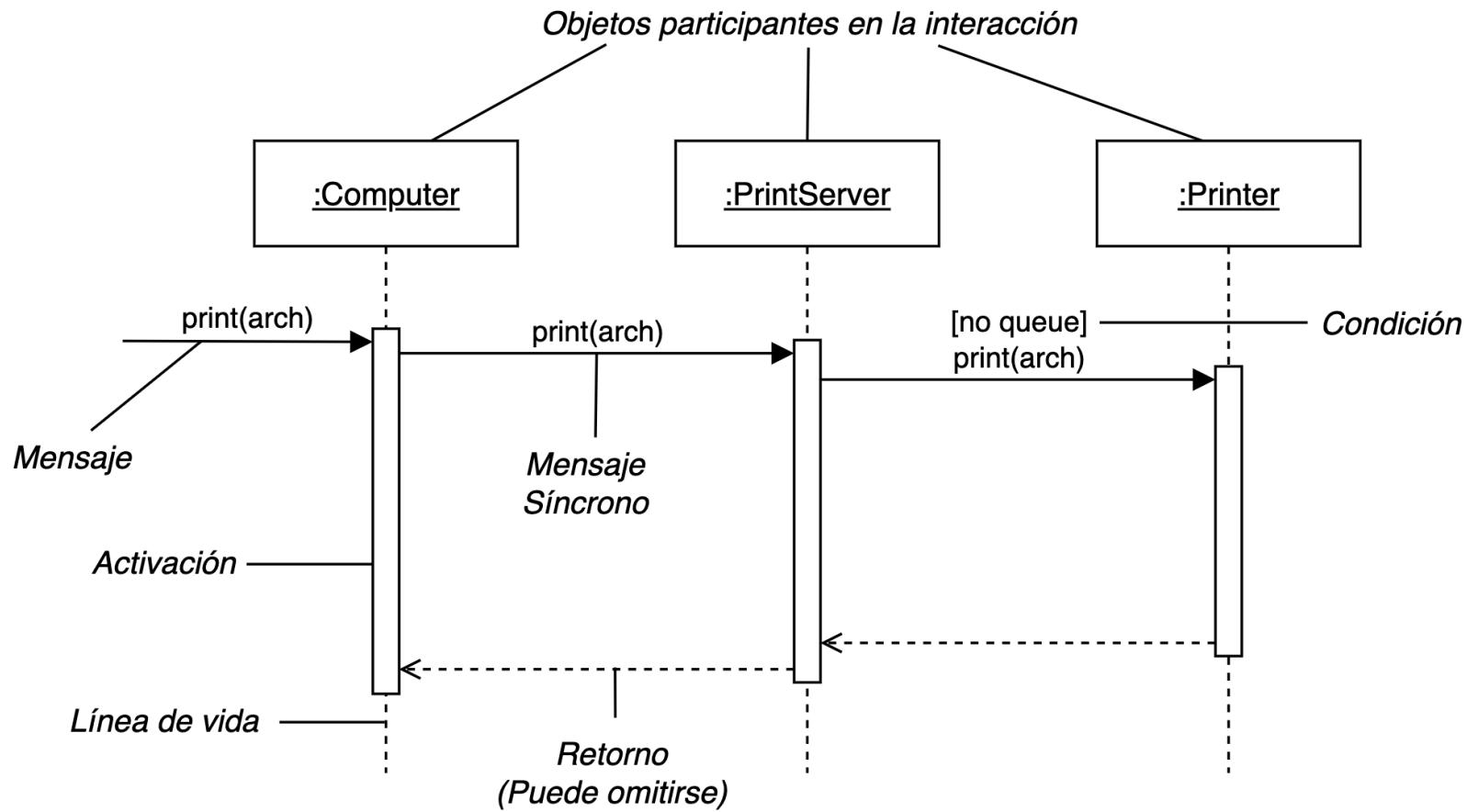
- Los diagramas de secuencia tienen dos características que los distinguen de los diagramas de colaboración
  - **Línea de vida:** Línea vertical discontinua que marca la duración de un objeto; comienza con el mensaje <<create>> y termina con <<destroy>>, añadiendo además una señal en X.
  - **Foco de control:** Rectángulo delgado y estrecho sobre la línea de vida, que muestra el tiempo de ejecución de una acción; permite anidamiento con rectángulos adicionales hacia la derecha



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

### Ejemplo



# El Modelo de Comportamiento en UML

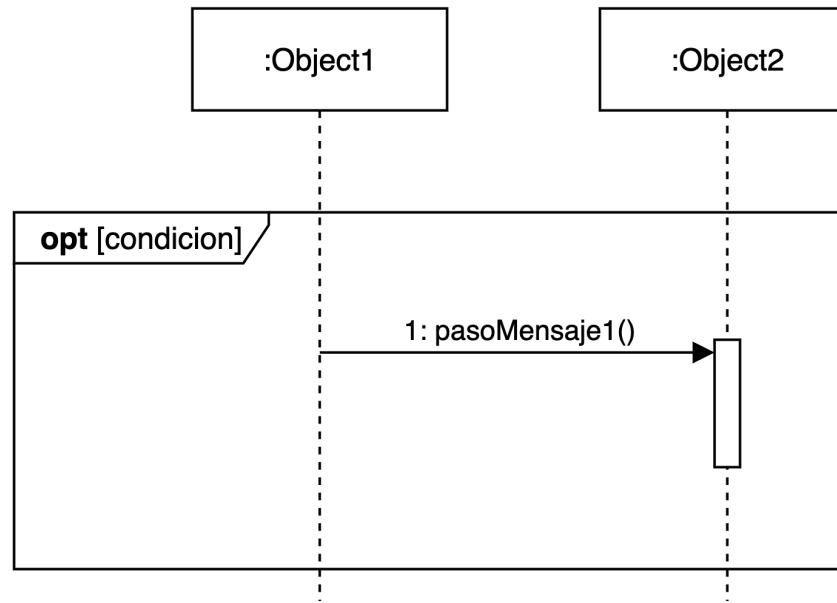
## Diagrama de Interacción – Diagrama de Secuencia

### Operadores de control y/o marcos de interacción

- Un **marco de interacción** es una sección de un diagrama de secuencia que contiene un operador de control para definir como se ejecuta la secuencia

#### Ejecución Opcional (opt)

Se ejecuta solo si se cumple una **condición**



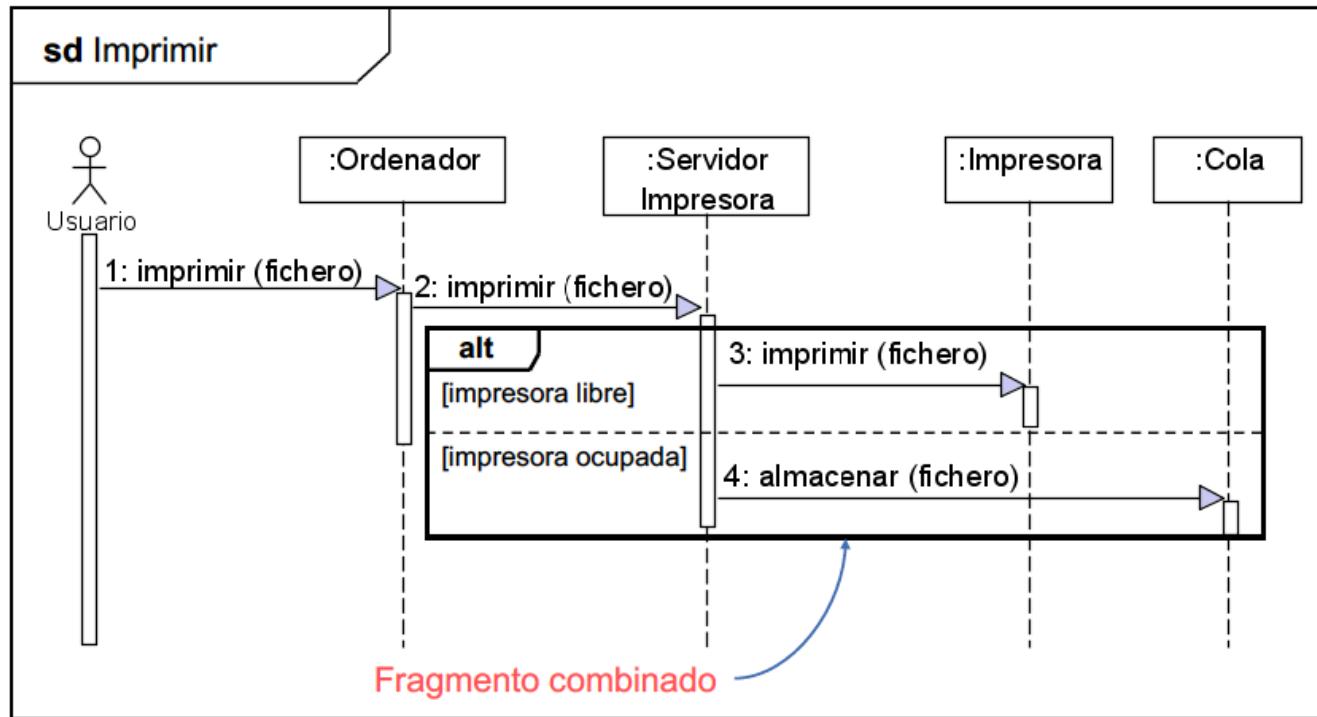
# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

### Operadores de control y/o marcos de interacción

#### Ejecución Condicional (alt)

El operador se divide en varias subregiones, cada una representando una **rama de la condición** con líneas discontinuas horizontales



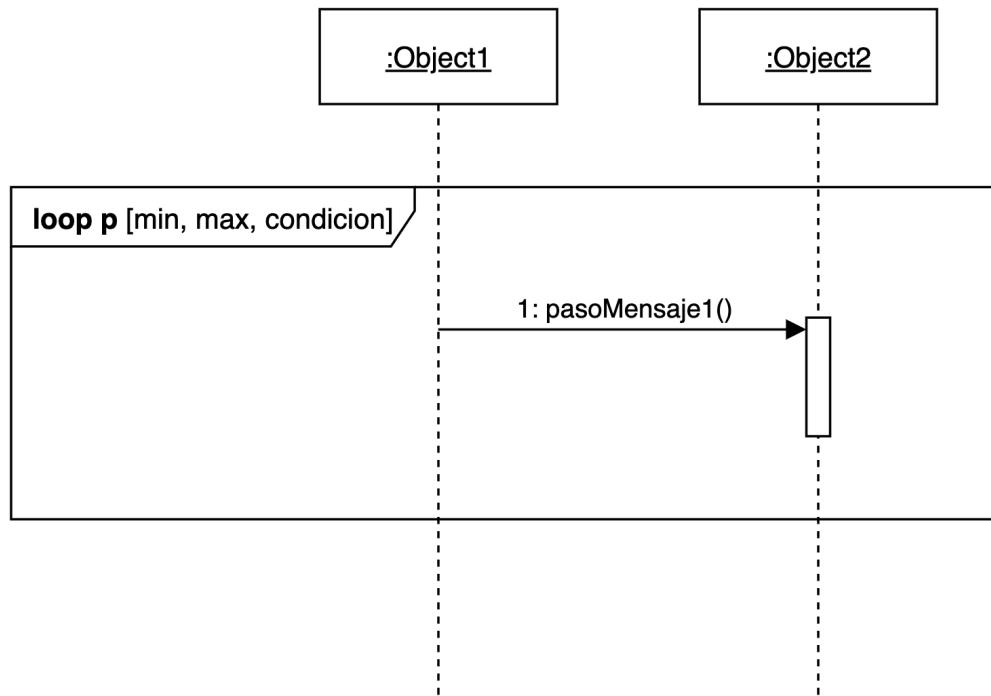
# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

### Operadores de control y/o marcos de interacción

#### Ejecución Iterativa (loop)

Comportamiento repetitivo, indicando min y max y la condición de guarda. El cuerpo del bucle se ejecuta mientras la condición de guarda sea cierta antes de cada iteración.



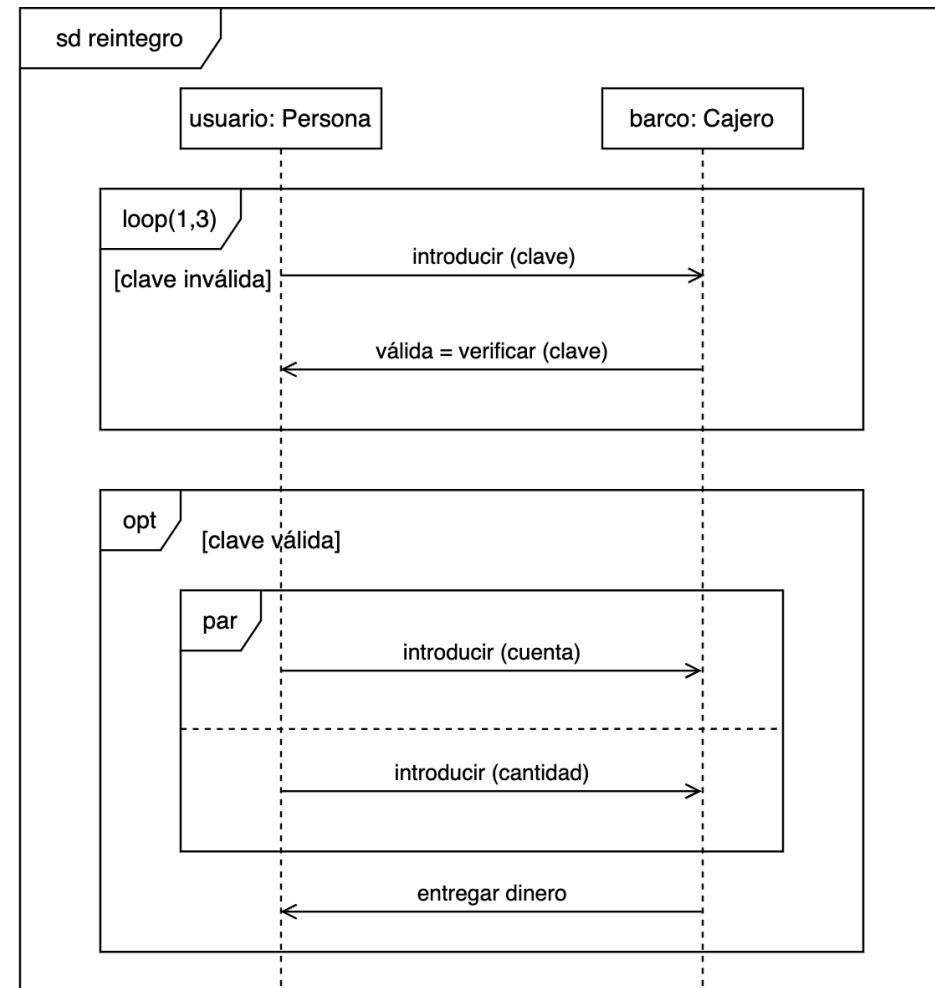
# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Secuencia

### Operadores de control y/o marcos de interacción

#### Ejecución Paralela (par)

El operador de control se divide en varias subregiones separadas por líneas discontinuas horizontales, cada una indicando una **computación paralela** (concurrente)



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
  1. Diagrama de Casos de Uso
  2. Diagrama de Actividades
  3. Diagrama de Máquinas de Estado
4. Diagrama de Interacción
  - Diagrama de Secuencia
  - Diagrama de Colaboración
3. El Modelo Estructural en UML
4. Técnicas Estructuradas



# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Colaboración

- Muestra la interacción espacial entre objetos, enfocándose en el paso de mensajes
- Utiliza los mismos elementos que los diagramas de secuencia, a excepción de las “líneas de vida” y “foco de control”
- Utilidad:
  - Identifica objetos y su relación en el sistema
  - Describir el flujo de mensajes entre los objetos o roles
- Resalta la organización estructural de los objetos que colaboran
- Representa objetos como nodos y enlaces como arcos etiquetados con los mensajes
- El orden de los mensajes se muestra con numeración decimal de Dewey (1, 1.1, ...)

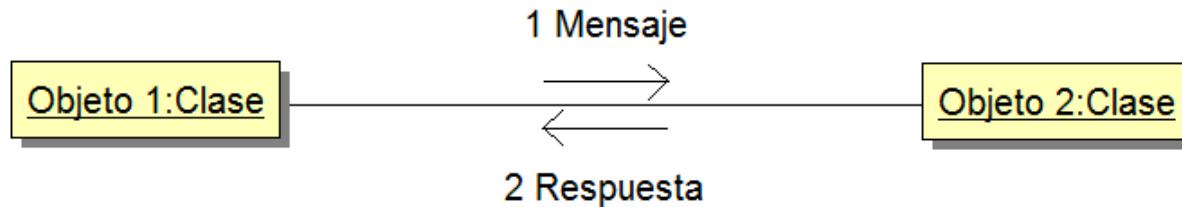


# El Modelo de Comportamiento en UML

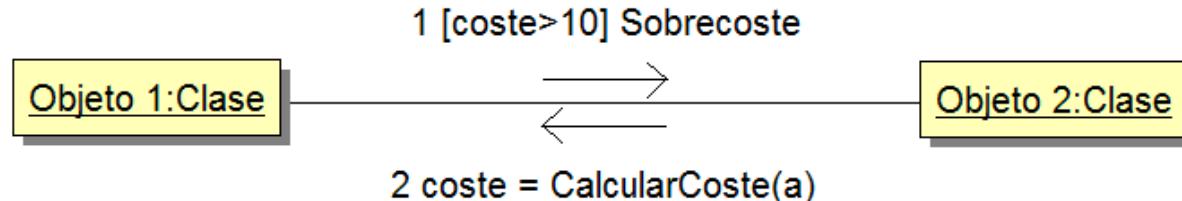
## Diagrama de Interacción – Diagrama de Colaboración

### Mensajes

- Cuando 2 objetos establecen una comunicación, se incluye un enlace, representado por una línea
- Los mensajes se muestran superpuestos al enlace
- El orden de ejecución de los mensajes se muestra junto a su texto descriptivo



- Un mensaje se puede expresar en lenguaje natural o en pseudocódigo, incluyendo condiciones o llamadas a funciones

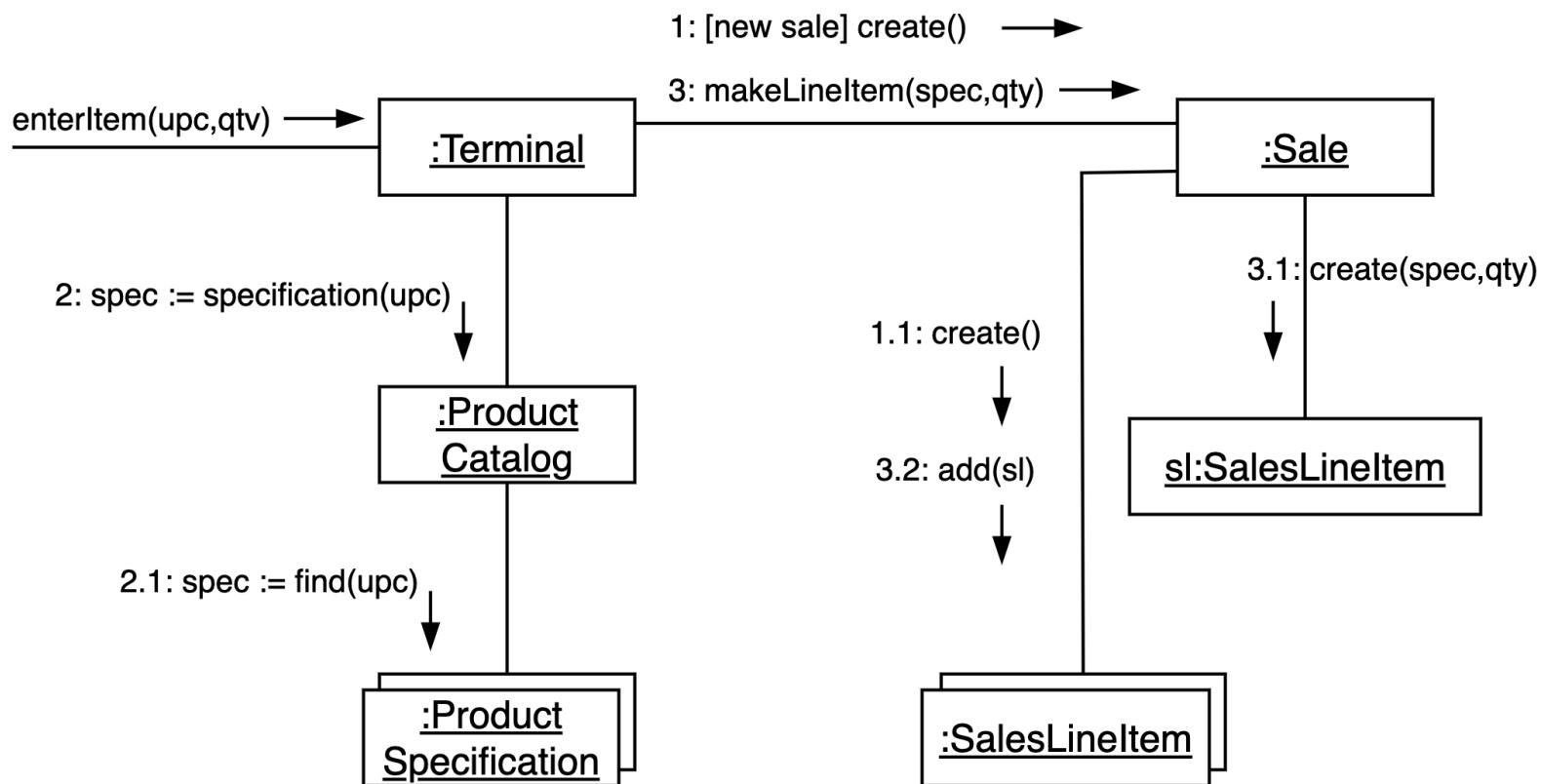


# El Modelo de Comportamiento en UML

## Diagrama de Interacción – Diagrama de Colaboración

### Ejemplo

- Diagrama de colaboración para la función “ingresar un ítem” en una aplicación de Punto-de-Venta



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. Diagrama de Paquetes
  4. Diagrama de Componentes
  5. Diagrama de Despliegue
4. Técnicas Estructuradas



# El Modelo Estructural en UML

## Introducción

- Las partes estáticas del sistema se representan por:
  - Diagrama de clases
  - Diagrama de objetos
  - Diagramas de paquetes
  - Diagrama de componentes
  - Diagrama de despliegue
- Los diagramas estructurales existen para visualizar, especificar, construir y documentar la estructura estable del sistema (“esqueleto”)
- Incluyen clases, interfaces, colaboraciones, componentes y nodos



# El Modelo Estructural en UML

## Diagrama de Clases

- Una **clase** define una categoría de objetos con características de estructura (atributos) y comportamiento (operaciones)
- Los **objetos** son instancias de una clase con valores específicos en sus atributos
- Las clases tienen **operaciones** que los objetos ejecutan al ser invocados
- Los **objetos se activan** al ser creados y permanecen activos hasta completar su función o ser terminados por otro objeto
- Los objetos activos responden a mensajes de otros objetos

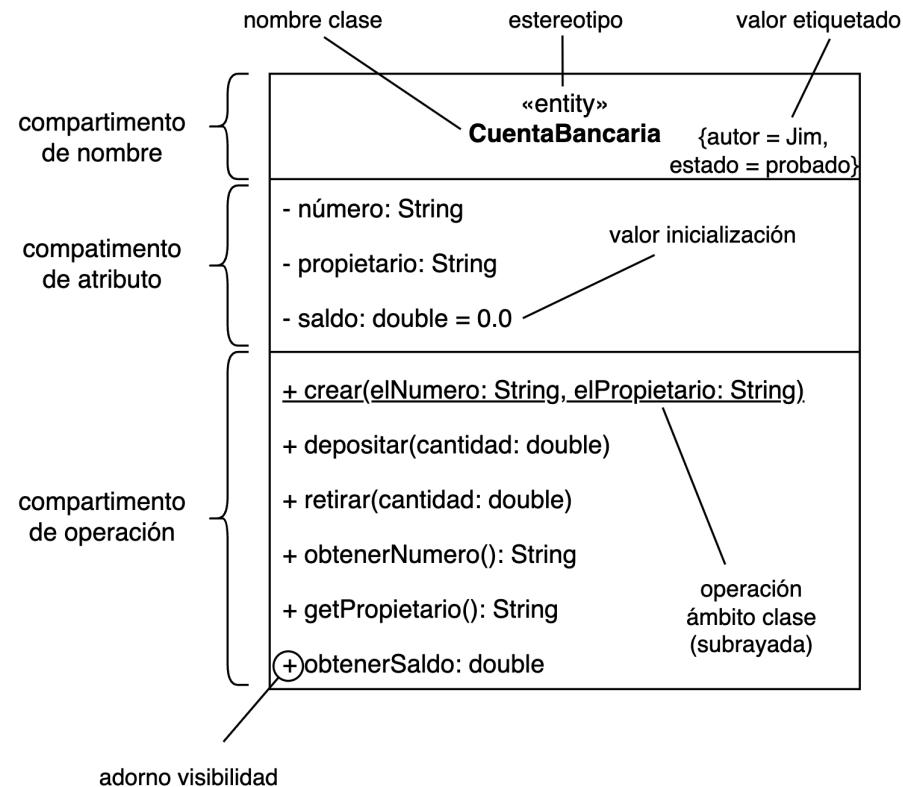


# El Modelo Estructural en UML

## Diagrama de Clases

- La especificación de una clase consta de las siguientes partes:

- Compartimento de nombre
  - Nombre de la clase
  - Estereotipos
  - Valores etiquetados
- Compartimento de atributo
  - Nombre de cada atributo
  - Visibilidad
  - Tipo
  - Valores de inicialización
- Compartimento de operación
  - Nombre de cada operación
  - Visibilidad
  - Tipo de retorno
  - Parámetros, incluyendo su tipo



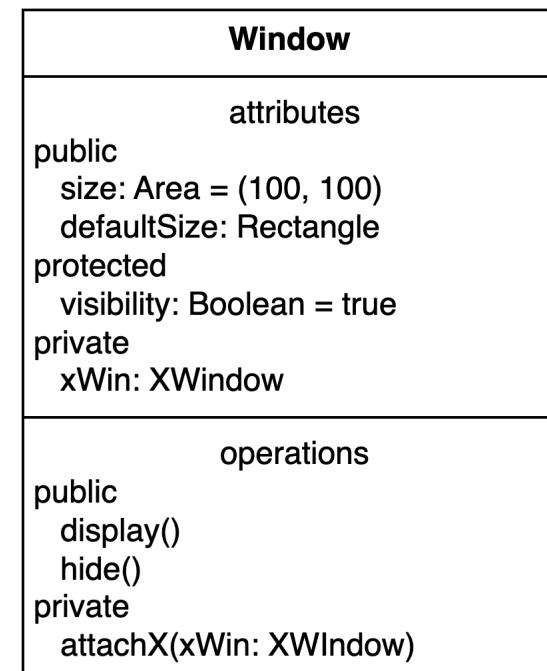
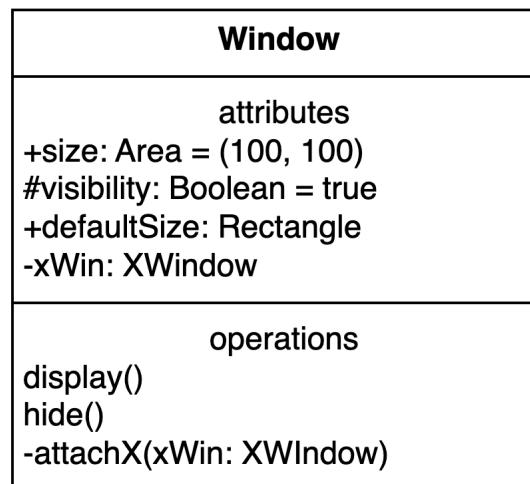
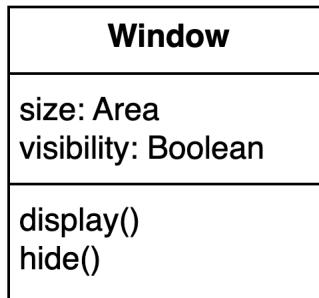
- Estos elementos se pueden simplificar y refinar durante el análisis y diseño, por lo que no siempre todos aparecen en el diagrama



# El Modelo Estructural en UML

## Diagrama de Clases

- Las clases pueden representarse de diferente manera, según el grado de detalle que se quiera establecer:



# El Modelo Estructural en UML

## Diagrama de Clases

### Clases

- **Compartimento de nombre**
  - Además del nombre (obligatoria), podemos indicar si la clase es abstracta (estereotipo <>abstract>>)
- **Compartimento de atributo**
  - Solo el nombre es obligatorio
  - Visibilidad: pública (+), privada (-), protegida (#), de paquete (~)
  - Tipo: UML soporta tipos primitivos (int, string, boolean) y enumeraciones
  - Multiplicidad: [0...1], [3], [2...\*]
  - Valor inicial: más habitual en la fase de diseño, salvo para especificar restricciones

Importante: No tiene por qué existir una correspondencia directa con la sintaxis y constructos de los lenguajes de programación

[visibilidad] nombre: tipo[multiplicidad] [= valor\_inicial]



# El Modelo Estructural en UML

## Diagrama de Clases

### Clases

- **Comportamiento de operación**
  - Solo el nombre es obligatorio (en análisis temprano)
  - Visibilidad: pública (+), privada (-), protegida (#), de paquete (~)
  - El tipo de retorno sigue las mismas normas respecto a los tipos soportados en UML
  - La lista de parámetros puede detallarse (más habitual en diseño):
    - Dirección del parámetro: in, out, inout
    - Tipo de parámetro
    - Valor predeterminado
  - Un ejemplo de propiedad es: {isQuery}

[visibilidad] nombre [(lista\_parámetros)] [: tipo\_retorno] [{propiedades}]

[dirección] nombre\_parámetro : tipo\_parámetro = valor\_predeterminado



# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

- Las relaciones son **conexiones semánticas significativas** entre elementos de un modelo
- En el ámbito del modelado de clases y objetos, podemos distinguir tres tipos: entre objetos (vínculos), entre clases (asociaciones y generalizaciones) y generales (dependencias)
- **Relación entre objetos:** conexión entre dos objetos para indicar que intercambian mensajes entre sí. Cuando se recibe un mensaje, se invoca a la operación correspondiente.
- **Relación de dependencia:** indica una relación entre dos o más elementos, donde un cambio en un elemento (el proveedor) puede afectar a otro elemento (el cliente)
  - Por ejemplo, para representar que una operación de una clase incluye un parámetro cuyo tipo es otra clase
  - Existen muchos estereotipos para indicar diferentes tipos de dependencias según la semántica que se le quiera dar



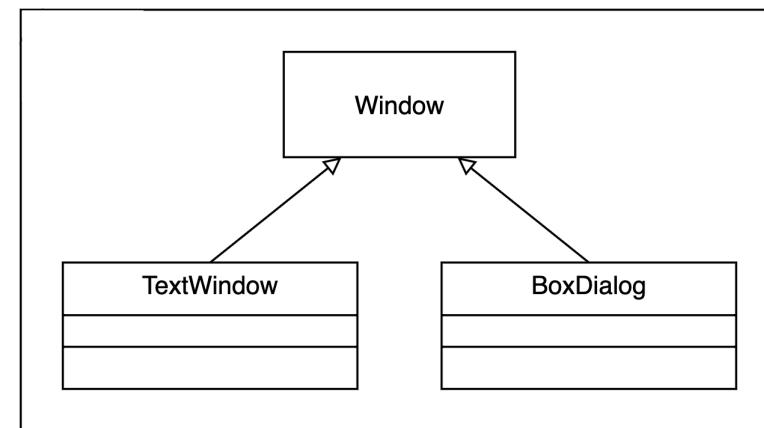
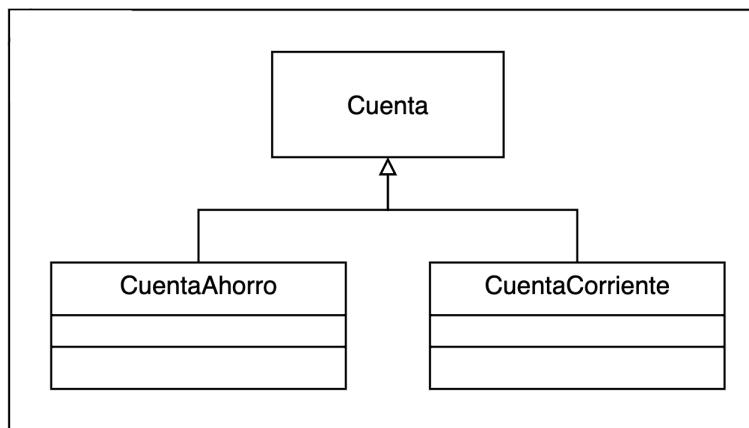
# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

#### Generalización

- Es una relación entre un elemento general y un tipo más específico de ese elemento
- La herencia simple es suficiente en la mayoría de los casos
- Se define como “es un tipo de”



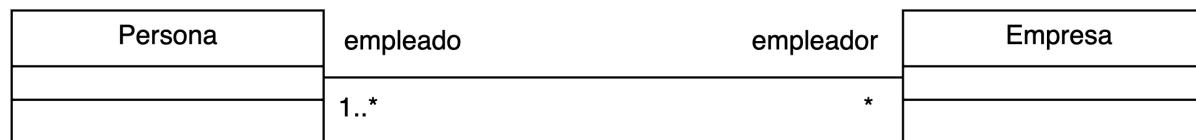
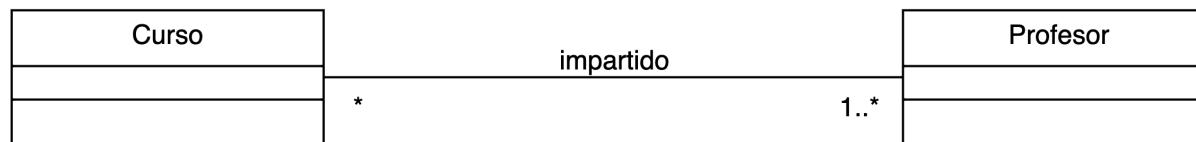
# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

#### Asociación

- Son relaciones entre clases, su semántica simplemente indica que pueden existir vínculos entre los objetos de ambas clases
- Del mismo modo que los objetos son instancias de las clases, los vínculos entre objetos son instancias de las asociaciones entre clases
- Las asociaciones pueden tener:
  - Un nombre, que indica la acción que el objeto fuente realiza sobre el destino
  - Nombres de roles (alternativa), para indicar qué “papel” desempeña cada objeto



# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

#### Asociación

- Las asociaciones pueden tener:

- Navegabilidad: Permite restringir la navegación a una sola dirección, indicando si una clase “conoce” a la otra
- Multiplicidad: Especifica cuántos objetos pueden estar involucrados en la relación en un momento dado, en uno o ambos extremos. Se define como un intervalo, con un mínimo y un máximo

Adorno	Semántica
0..1	Cero o uno
1	Exactamente uno
0..*	Cero o más
*	Cero o más
1..*	Uno o más
n...m	Entre n y m



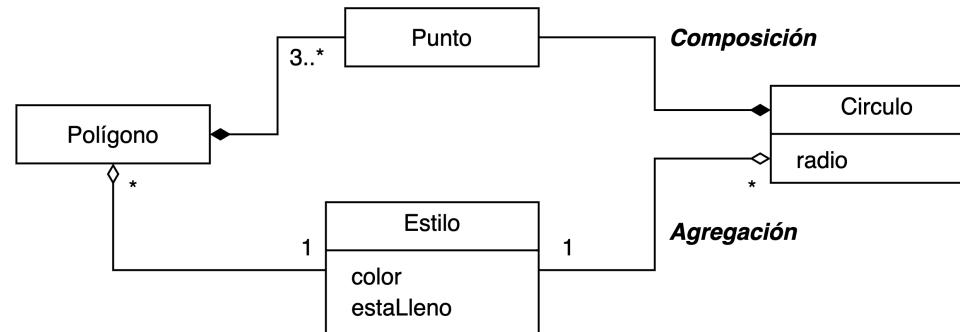
# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

#### Agregación y composición

- Son tipos especiales de asociación
- **Agregación:** relación entre una clase que constituye un “todo” y otras clases que representan sus partes. Cada clase puede existir independientemente (p.ej. Ordenador y sus periféricos)
- **Composición:** agregación fuerte entre las partes y el todo. La diferencia es que las partes no tienen vida independiente fuera del todo
- En la composición, las partes pertenecen a un único “todo”. En la agregación pueden ser compartidos entre varios elementos



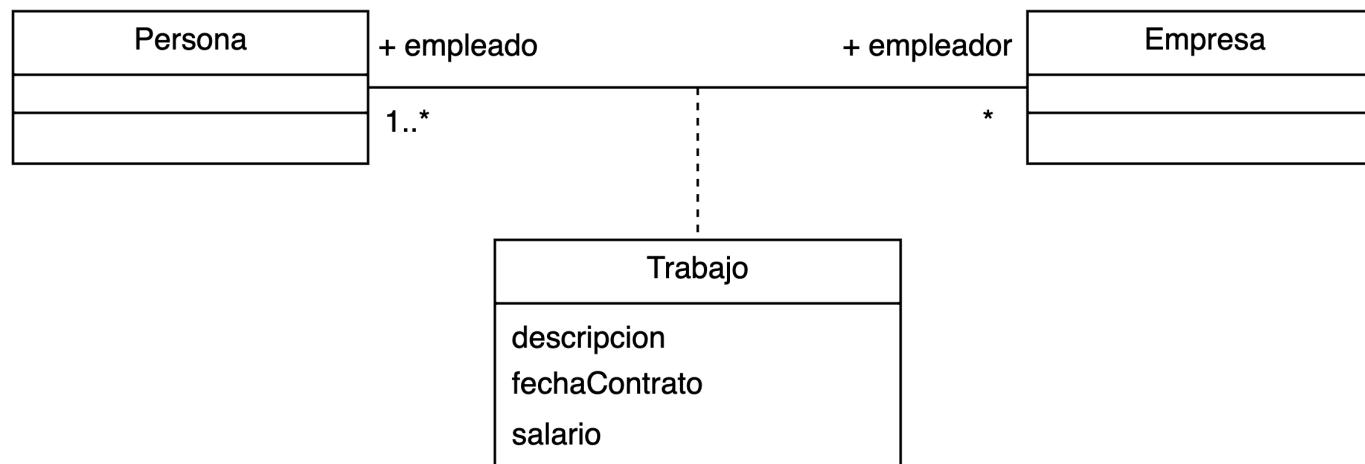
# El Modelo Estructural en UML

## Diagrama de Clases

### Relaciones

#### Clase asociación

- Útiles en relaciones de asociación con multiplicidad \*, cuando los atributos no encajan en ninguna de las dos clases.
- Forma parte de la asociación, junto a todas sus propiedades.
- Es a la vez una clase que puede tener operaciones y atributos.
- Ejemplo: No podríamos modelar que una persona tiene diferentes contratos para una misma compañía a lo largo del tiempo.



# El Modelo Estructural en UML

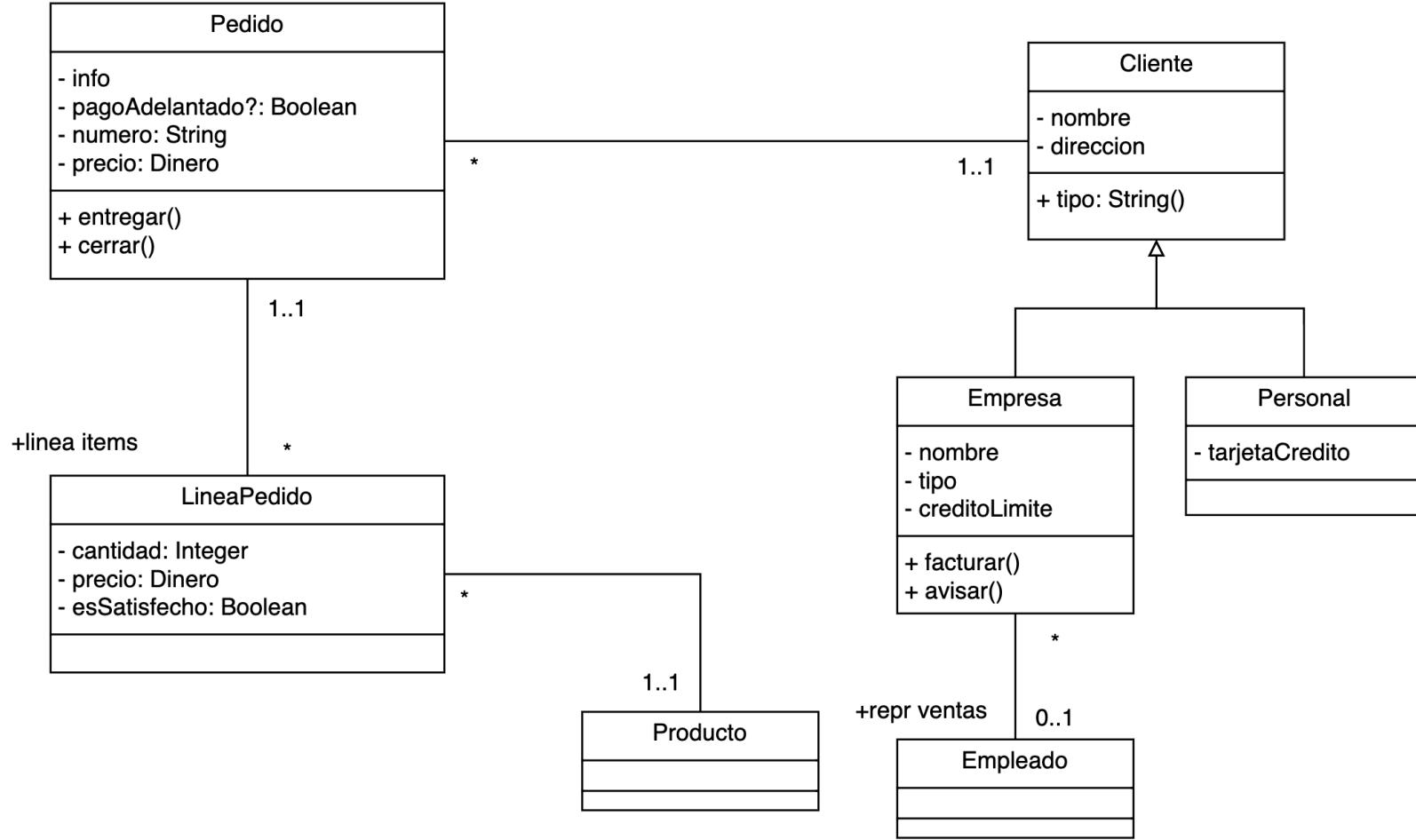
## Diagrama de Clases

- Son los más utilizados en el modelado orientado a objetos y nos muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones
- Están basados en la consideración de que el mundo real puede ser visto desde diferentes abstracciones (subjetividad), existiendo diferentes mecanismos de abstracción como
  - Clasificación / Instanciación
  - Composición / Descomposición
  - Agrupación / Individualización
  - Especialización / Generalización
- La clasificación es uno de los mecanismos de abstracción más utilizados



# El Modelo Estructural en UML

## Diagrama de Clases



# El Modelo Estructural en UML

## Diagrama de Clases

- Perspectivas de los Diagramas de Clases (Cook y Daniels, 1994)
  - **Conceptual:** Representa los conceptos del dominio, sin relación directa con la implementación y es independiente del lenguaje
  - **Especificación:** Define tipos de interfaces que pueden tener distintas implementaciones, dependiendo del entorno, desempeño o proveedor
  - **Implementación:** Muestra las clases tal como se implementarán en el código. Es la más común, aunque a veces es mejor usar la perspectiva de especificación



# El Modelo Estructural en UML

## Diagrama de Clases

### Comentarios sobre el modelado

- Considerar casos de uso y escenarios para descubrir relaciones entre abstracciones, evitando modelar de forma aislada.
- Pasos:
  1. Comenzar con relaciones estructurales (vista estática y tangible del sistema)
  2. Identificar generalizaciones/especializaciones (herencia múltiple moderada)
  3. Agregar dependencias para una vista semántica
- Partir de relaciones básicas y agregar detalles solo cuando sea necesario para clarificar la intención
- Modelar relaciones complejas de manera incremental



# El Modelo Estructural en UML

## Diagrama de Clases

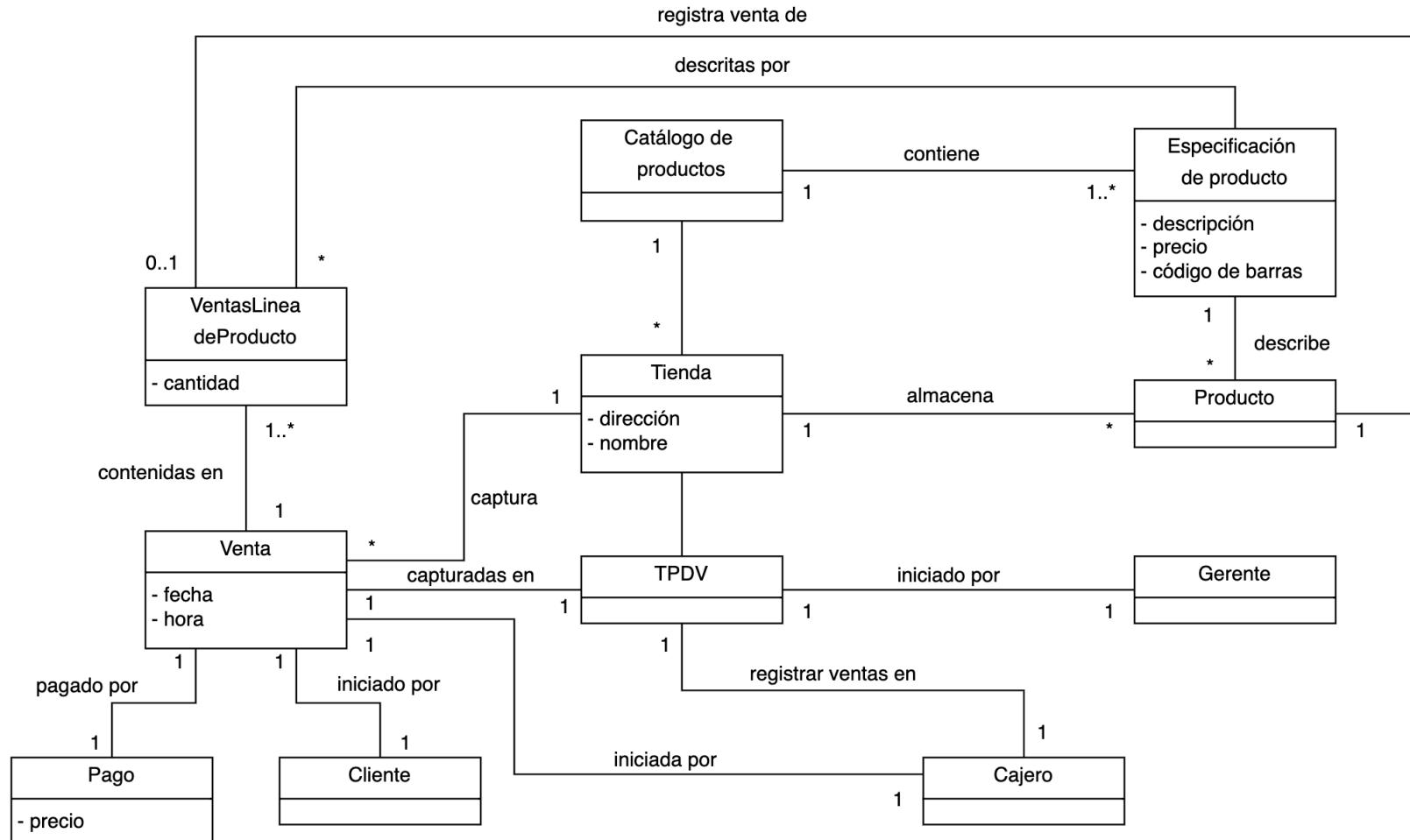
### Sugerencias y consejos

- Un diagrama de clases es una representación gráfica de la vista de diseño estática de un sistema, pero no necesita ser completo por sí solo. Varios diagramas juntos proporcionan una visión integral del sistema
- Un diagrama de clases bien estructurado debe:
  - Centrarse en comunicar un aspecto específico del sistema
  - Incluir solo los elementos esenciales para comprender ese aspecto
  - Proporcionar detalles adecuados al nivel de abstracción
  - Tener un nombre que indique claramente su propósito
  - Evitar mostrar demasiados tipos de relaciones



# El Modelo Estructural en UML

## Diagrama de Clases



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. Diagrama de Paquetes
  4. Diagrama de Componentes
  5. Diagrama de Despliegue
4. Técnicas Estructuradas



# El Modelo Estructural en UML

## Diagrama de Objetos

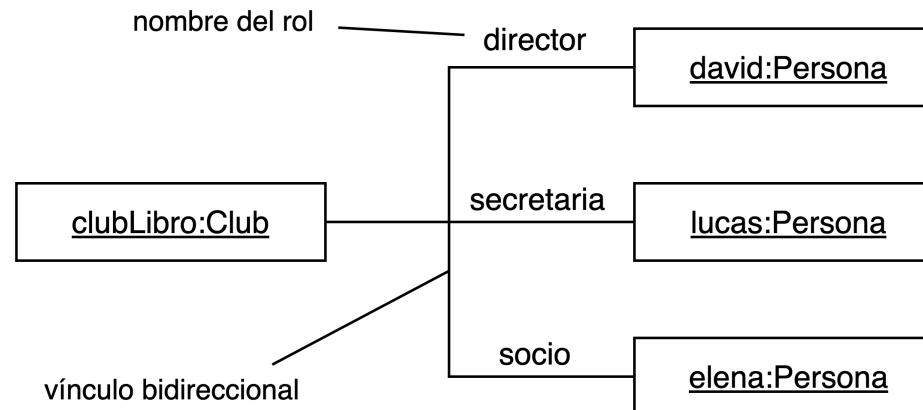
- Un diagrama de objeto muestra una vista completa o parcial de los objetos **en un instante de ejecución específico**
- Su objetivo es modelar las instancias de los elementos presentes en los diagramas de clases
- Características:
  - Modela la vista estática de diseño y procesos del sistema
  - Cada instancia debe tener un nombre único dentro de su contexto
  - Las operaciones que pueden ejecutarse en el objeto se declaran en su abstracción
  - Comparte notación con los diagramas de clases. El nombre del objeto se subraya
- Utilidad:
  - Ilustrar estructuras de datos/objetos del sistema
  - Especificar detalles del modelo
  - Obtener una “foto” del sistema en un momento específico



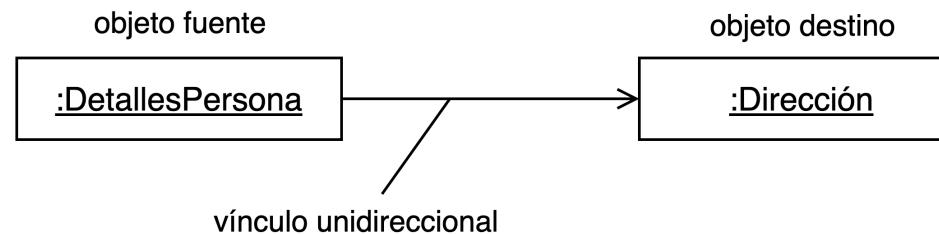
# El Modelo Estructural en UML

## Diagrama de Objetos

- Los objetos pueden tener roles si están conectados por vínculos
  - Vínculo Bidireccional:** ambos elementos tienen conocimiento mutuo y pueden interactuar entre sí



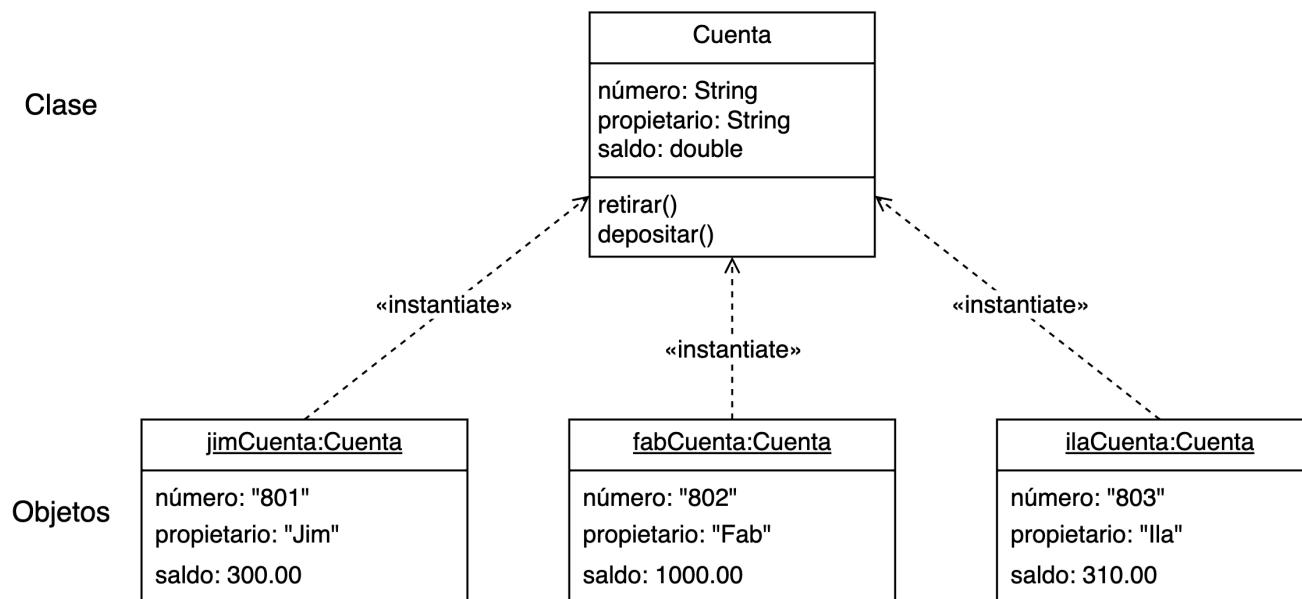
- Vínculo Unidireccional:** solo uno de los elementos tiene conocimiento del otro y puede interactuar con él, pero no al revés



# El Modelo Estructural en UML

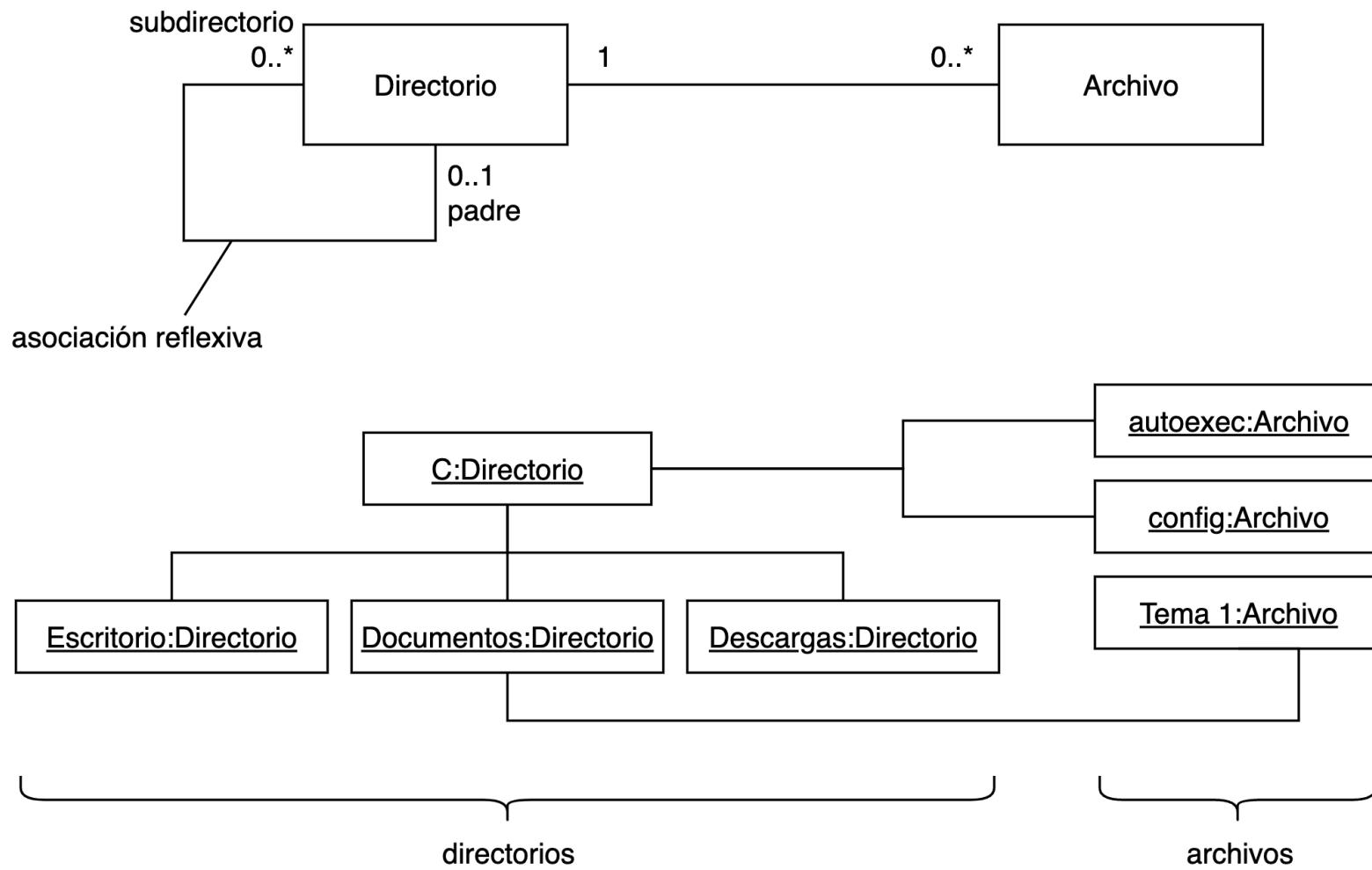
## Diagrama de Objetos

- Al visualizar el estado de un objeto, se muestra el valor de sus atributos en un momento específico, ya que el estado es dinámico
- Existen dos estereotipos en las relaciones de dependencia entre objetos y clases:
  - instanceOf**: indica que el objeto es una instancia de la clase especificada
  - instantiate**: indica que la clase crea instancias de otra clase



# El Modelo Estructural en UML

## Diagrama de Objetos



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. **El Modelo Estructural en UML**
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. **Diagrama de Paquetes**
  4. Diagrama de Componentes
  5. Diagrama de Despliegue
4. Técnicas Estructuradas



# El Modelo Estructural en UML

## Diagrama de Paquetes

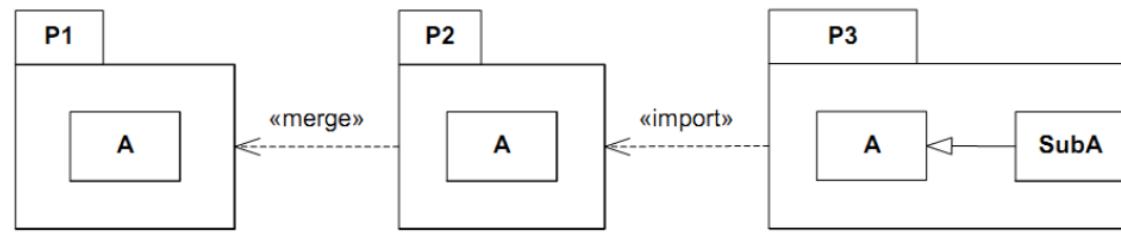
- Los paquetes se utilizan para organizar los elementos de modelado en partes mayores que se pueden manipular como un grupo
- Los paquetes constituyen un mecanismo de agrupación para organizar elementos UML
- La visibilidad de los elementos debe controlarse para que algunos sean visibles fuera del paquete y otros permanezcan ocultos
- Los elementos incluidos en el mismo paquete suelen ser cercanos semánticamente y suelen cambiar juntos, entonces diremos que un paquete es bien estructurado y que es cohesivo y poco acoplado, estando controlado el acceso a su contenido
- Gráficamente un paquete se representa como una carpeta y ha de tener un nombre que lo distinga de otros, que puede ser
  - Nombre simple solo una cadena de texto
  - Nombre de camino, nombre de paquete precedido por el nombre del paquete contenedor



# El Modelo Estructural en UML

## Diagrama de Paquetes

- Un paquete puede contener otros elementos, incluyendo clases, interfaces, componentes, nodos, colaboraciones, casos de uso, diagramas y otros paquetes
- Cada elemento pertenece exclusivamente a un único paquete
- Se puede controlar la visibilidad de los elementos contenidos en un paquete de la misma manera en que lo hacemos en las clases
  - Público, visible a los contenidos de cualquier paquete que importe al paquete contenedor del elemento
  - Privado, no son visibles fuera del paquete en el que se declaran
- Los paquetes pueden estar sujetos a relaciones de generalización y dependencias, estas últimas con semántica propia (merge, import)



# El Modelo Estructural en UML

## Diagrama de Paquetes

### Importación

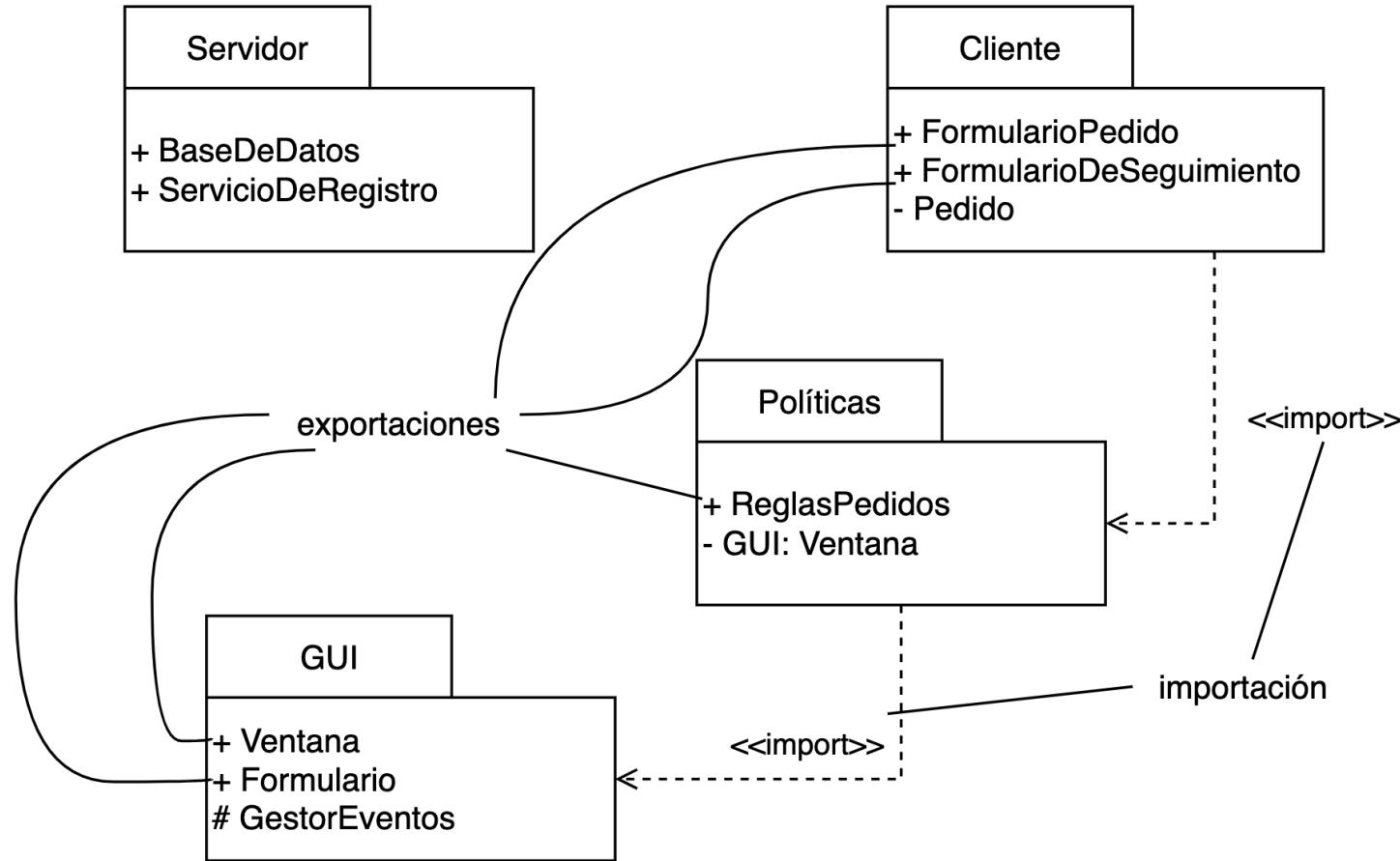
- La importación concede un permiso de un solo sentido para que los elementos de un paquete accedan a los elementos de otro
- La relación de importación en UML se modela como una dependencia con el estereotipo *import*
- Las partes públicas de un paquete son sus exportaciones
- Las partes que exporta un paquete son sólo visibles al contenido de aquellos paquetes que lo importan explícitamente
- Las dependencias en los paquetes “no son transitivas”, permitiendo el propósito de la arquitectura por capas
- Si un elemento es visible en un paquete es visible en todos los paquetes incluidos en ese paquete
- Los paquetes anidados pueden ver todo lo que los paquetes que lo contienen



# El Modelo Estructural en UML

## Diagrama de Paquetes

### Importación

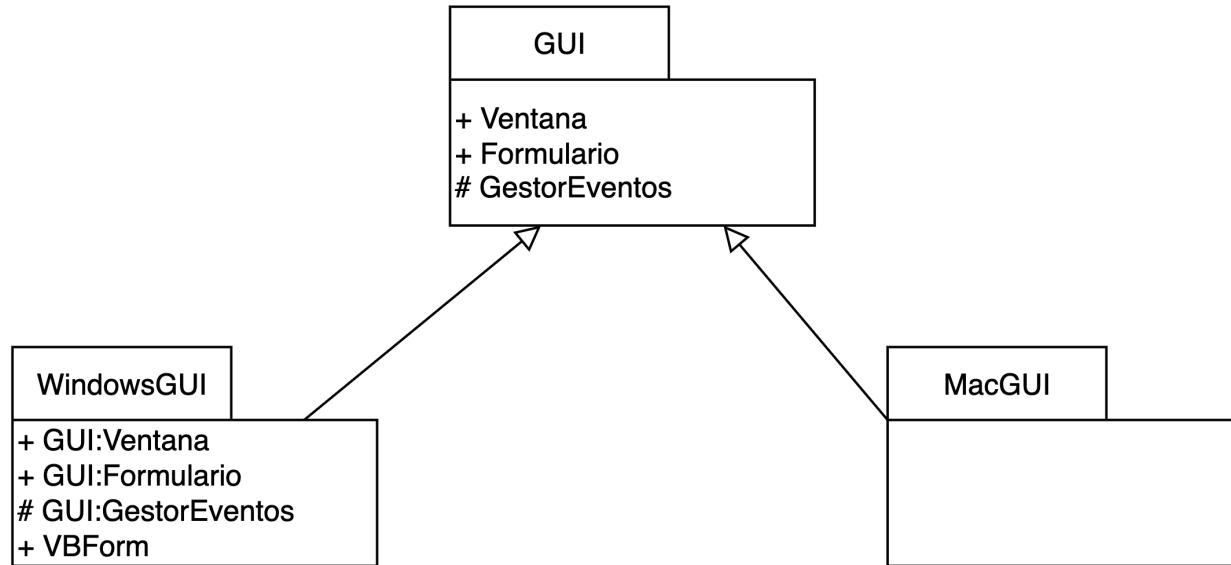


# El Modelo Estructural en UML

## Diagrama de Paquetes

### Generalización

- Es el otro tipo de relación que se puede dar entre paquetes y nos sirve para especificar las familias de paquetes
- Los paquetes implicados en la relación siguen el mismo principio de sustitución de las clases



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. **El Modelo Estructural en UML**
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. Diagrama de Paquetes
  4. **Diagrama de Componentes**
  5. Diagrama de Despliegue
4. Técnicas Estructuradas



# El Modelo Estructural en UML

## Diagrama de Componentes

- El diagrama de componentes muestra como un sistema se divide en **componentes**, así como las relaciones entre ellos
- Poseen un nivel de abstracción superior a los diagramas de clases, ya que usualmente un componente se implementa por una o más clases en tiempo de ejecución
- Utilizados en su mayor parte en el ámbito de la arquitectura del software
- Utilidad
  - Modelar la vista lógica de un sistema
  - Modelar el código fuente
  - Modelar las diferentes versiones ejecutables
  - Modelar bases de datos físicas
  - Modelar sistemas adaptables



# El Modelo Estructural en UML

## Diagrama de Componentes

- Un componente es una unidad autónoma en un sistema, definida por interfaces
- Su objetivo es encapsular parte de las funcionalidades del sistema, accesibles solo a través de sus interfaces
- El componente puede ser sustituido por otro que cumpla la misma especificación (autocontenido y reemplazable)
- Su estructura interna puede modelarse para especificar la delegación de la realización de las interfaces a alguna de sus partes internas.
- Los componentes pueden depender de otros componentes, lo que en realidad significa que depende de las interfaces de otros.
- Pueden ir estereotipados para enriquecer la semántica de lo que representan, por ejemplo, <<subsystem>>
- Tipos de componentes
  - Ejecutables: Se ejecutan de forma autónoma
  - Librerías: Biblioteca de objetos estática o dinámica
  - Tabla: Tabla en una Base de Datos
  - Archivo: Contiene un código fuente o datos
  - Documento: Otro tipo de documento



# El Modelo Estructural en UML

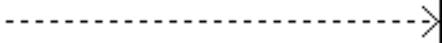
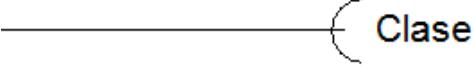
## Diagrama de Componentes

- El otro elemento fundamental del modelo arquitectónico son las interfaces
- Las interfaces especifican un conjunto de características públicas (normalmente servicios) que otros pueden utilizar
- La idea fundamental es separar la especificación (qué) de la implementación (cómo)
- Una misma interfaz puede “realizarse” de muchas maneras, y todas ellas son válidas mientras se cumplan las condiciones impuestas en la interfaz
- Los atributos y operaciones de la interfaz deben estar completamente especificados (“contrato”):
  - Signatura completa de la operación (nombre, parámetros y retorno)
  - Semántica de la operación (texto o pseudocódigo)
  - Nombre y tipo de los atributos
  - Restricciones



# El Modelo Estructural en UML

## Diagrama de Componentes

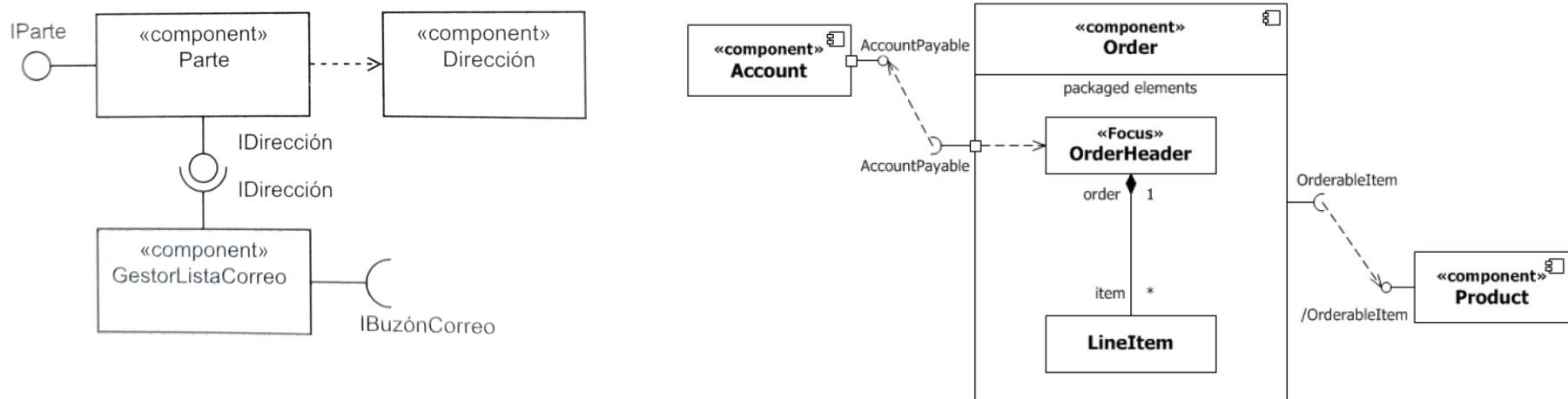
Relaciones	Notación
Dependencia	
Generalización (Herencia)	
Proporciona (Interfaz)	
Consumo (Interfaz)	



# El Modelo Estructural en UML

## Diagrama de Componentes

- Muestra cómo se interrelacionan los componentes, por medio de sus interfaces, para obtener la arquitectura completa de un sistema.
- Pueden mostrar, o no, la composición interna de los componentes o la especificación completa de las interfaces
- Mediante este tipo de diagramas se pueden representar patrones arquitectónicos como la arquitectura en capas



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. **El Modelo Estructural en UML**
  1. Diagrama de Clases
  2. Diagrama de Objetos
  3. Diagrama de Paquetes
  4. Diagrama de Componentes
  5. Diagrama de Despliegue
4. Técnicas Estructuradas



# El Modelo Estructural en UML

## Diagrama de Despliegue

- El diagrama de despliegue muestra la topología hardware del sistema
- Captura la relación entre los elementos de un modelo conceptual o físico, y los recursos de información asignados a ellos
- Utilizados en su mayor parte en el ámbito de la arquitectura. Desarrollado por diseñadores, ingenieros de sistemas e ingenieros de redes
- Utilidad
  - Indicar la distribución de los componentes
  - Evaluar el rendimiento y la carga del hardware del sistema
  - Examinar redundancia, balance de carga, etc

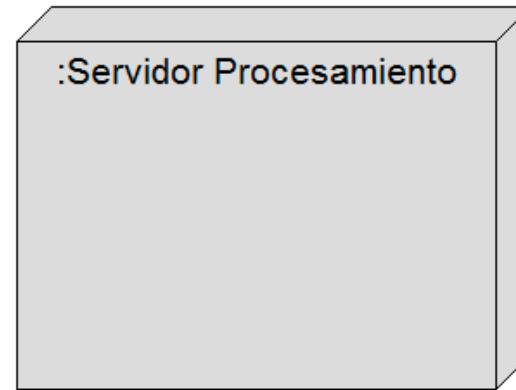


# El Modelo Estructural en UML

## Diagrama de Despliegue

### Nodos

- Los nodos son un elemento físico en tiempo de ejecución que representa un recurso computacional, como un procesador o dispositivo
- Puede contener objetos o instancias
- Modelan la topología de hardware donde se ejecuta el sistema
- Debe tener un nombre único (simple o con ruta de paquete)
- Los nodos también pueden incluir valores etiquetados o compartimentos para mostrar información extra

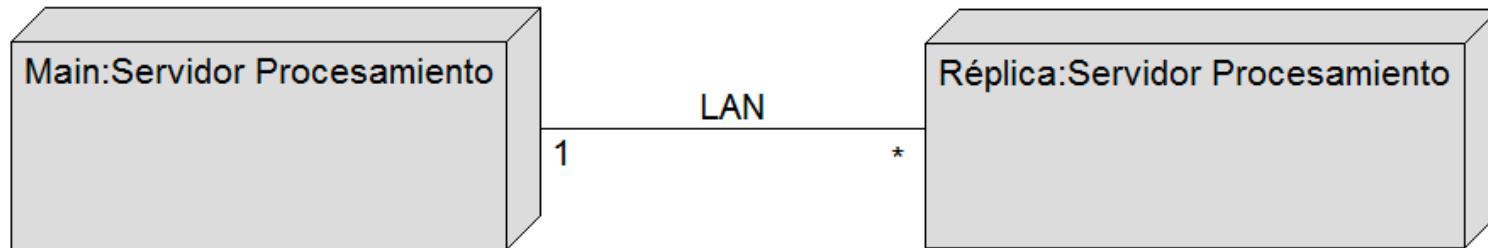


# El Modelo Estructural en UML

## Diagrama de Despliegue

### Relaciones

- Une los diferentes componentes del diagrama de despliegue
- En una relación se puede representar
  - El tipo de comunicación entre componentes, a través de una etiqueta
  - Cardinalidad de la relación



# El Modelo Estructural en UML

## Diagrama de Despliegue

### Artefactos

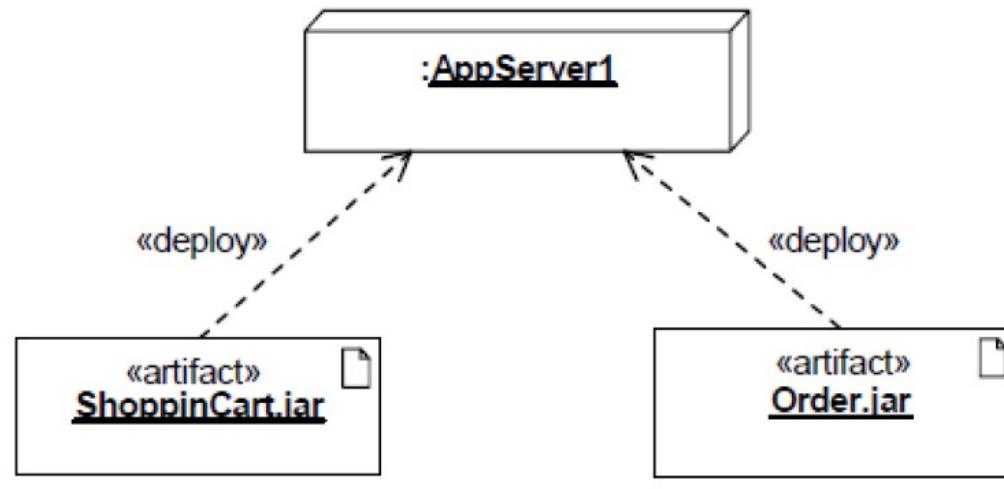
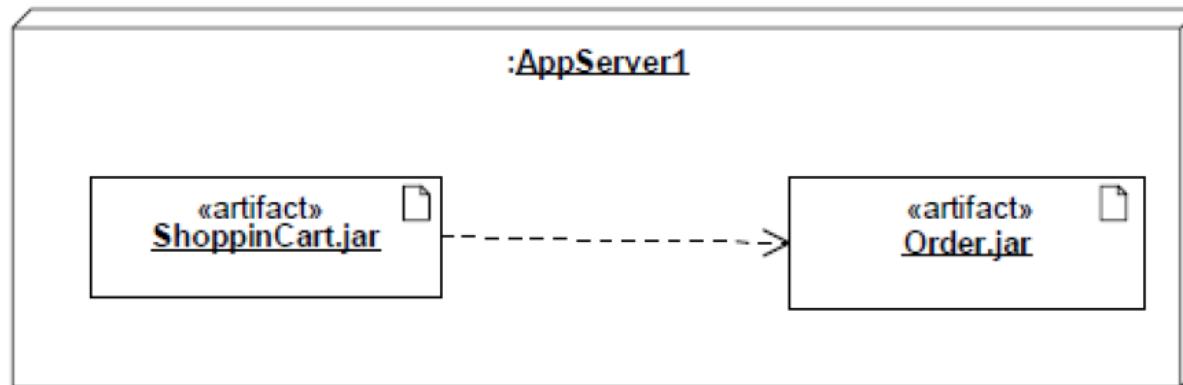
- Los artefactos especifican elementos de implementación
- Los artefactos se pueden situar
  - Dentro de los nodos: Especifican el recurso computacional donde se ejecutarán
  - Mediante relaciones: Sin especificar el recurso de ejecución
- Los artefactos son los elementos que se despliegan en los nodos, y los podemos asociar a: Archivos fuente, ejecutables, librerías, scripts, tablas de base de datos, documentos
- Estereotipos:
  - Se usa <<artifact>> para indicar un artefacto general
  - Es posible especificar más con <<script>>, <<executable>>, <<library>>, <<document>>, etc



# El Modelo Estructural en UML

## Diagrama de Despliegue

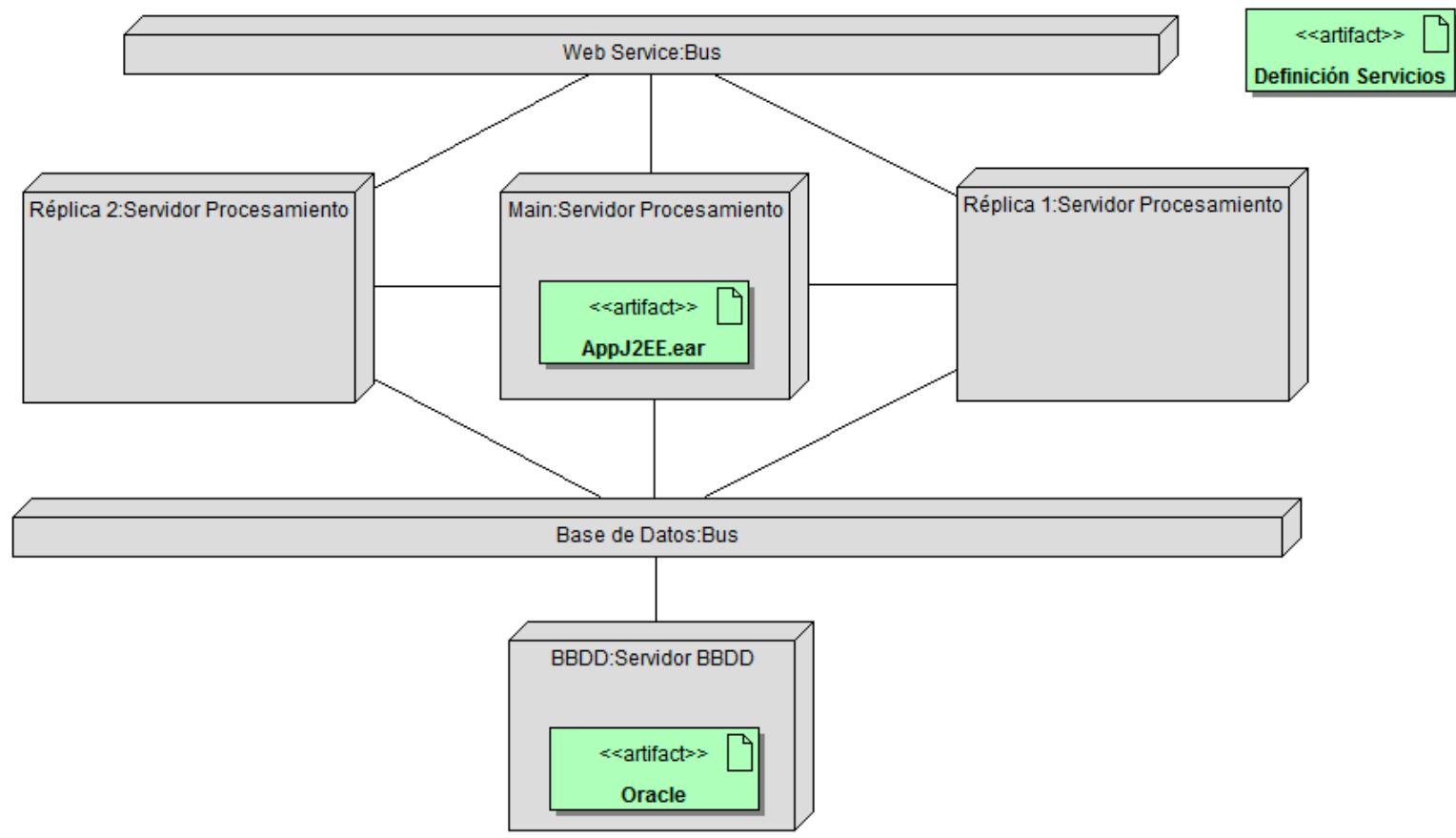
### Notación



# El Modelo Estructural en UML

## Diagrama de Despliegue

### Ejemplo



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. **Técnicas Estructuradas**
  1. Clasificación según el Enfoque de Representación
  2. Clasificación según el Enfoque de Modelado



# Técnicas Estructuradas

- Según Yourdon no existe un criterio definitivo por el que podamos organizar las técnicas de la metodología estructurada. Aunque si podemos considerar dos enfoques:
- Según el enfoque de **representación**, clasifica las técnicas según la forma en que se representan los componentes y aspectos del sistema
- Según el enfoque de **modelado**, clasifica las técnicas en función de los modelos del sistema que se crean



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. Técnicas Estructuradas
  1. Clasificación según el Enfoque de Representación
  2. Clasificación según el Enfoque de Modelado



# Técnicas Estructuradas

## Clasificación según el Enfoque de Representación

- El enfoque de representación organiza las técnicas según cómo presentan los elementos del sistema, permitiendo distintos niveles de detalle
  - **Gráficas:** Usan símbolos para mostrar visualmente componentes específicos, como funciones o flujos de datos, combinándose con otras técnicas para ofrecer una visión completa
  - **Textuales:** Describen en detalle los componentes de los gráficos usando un lenguaje formal, para mayor precisión
  - **Marcos o Plantillas:** Organizan información de los componentes ya descritos, documentando sus propiedades de manera estructurada
  - **Matriciales:** Verifican la precisión y consistencia de los modelos, mostrando referencias cruzadas entre componentes



# Índice

1. Introducción al modelado de software y UML
2. El Modelo de Comportamiento en UML
3. El Modelo Estructural en UML
4. Técnicas Estructuradas
  1. Clasificación según el Enfoque de Representación
  2. Clasificación según el Enfoque de Modelado



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

- El enfoque de modelado organiza las técnicas según el tipo de modelo creado:
  - **Modelado basado en la Información**
  - **Modelado basado en el Tiempo**
  - **Modelado basado en la Función**



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Modelado basado en la Información

- Este enfoque representa las entidades del sistema y sus relaciones, describiendo los datos que maneja, su transformación y salida. Su objetivo es capturar el dominio de datos del sistema, detallando los datos que se aceptan, procesan y producen
- Técnicas principales:
  - Diagramas Entidad-Relación (ER): Representan entidades y relaciones
  - Diagramas de Estructura de Datos: Muestran la organización interna de los datos
  - Matriz Entidad/Entidad: Organiza las relaciones entre entidades



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Modelado basado en el Tiempo

- Este enfoque describe cómo el sistema responde a eventos específicos, enfocándose en cuándo se activan funciones o procesos en respuesta a estos
- Objetivo: definir la secuencia de acciones activadas por eventos en el sistema
- Técnicas principales:
  - Diagramas de Transición de Estados: Muestran estados del sistema y transiciones según eventos
  - Listas de Eventos: Enumeran eventos que desencadenan respuestas
  - Redes de Petri: Modelan secuencias y paralelismos de eventos
  - Diagramas de Historia de Vida: Describen el ciclo de vida de entidades según eventos
  - Matriz Evento/Entidad: Relaciona eventos con entidades dependientes



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Modelado basado en la Función

- Este enfoque representa las funciones y procesos del sistema y cómo interactúan con los datos, detallando las tareas y transformaciones que realiza
- Objetivo: especificar los procesos y operaciones del sistema y cómo manejan la información
- Técnicas principales:
  - Diagramas de Flujo de Datos (DFD): Muestran funciones, entradas, salidas y sus interfaces
  - Lenguaje Estructurado: Define procesos o funciones con mayor precisión
  - Árboles y Tablas de Decisión: Representan decisiones complejas y reglas de decisión
  - Diagramas de Descomposición Funcional: Dividen el sistema en subfunciones
  - Matriz Función/Entidad: Relaciona funciones con entidades involucradas
  - Diccionario de Datos: Define y documenta todos los elementos de datos



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

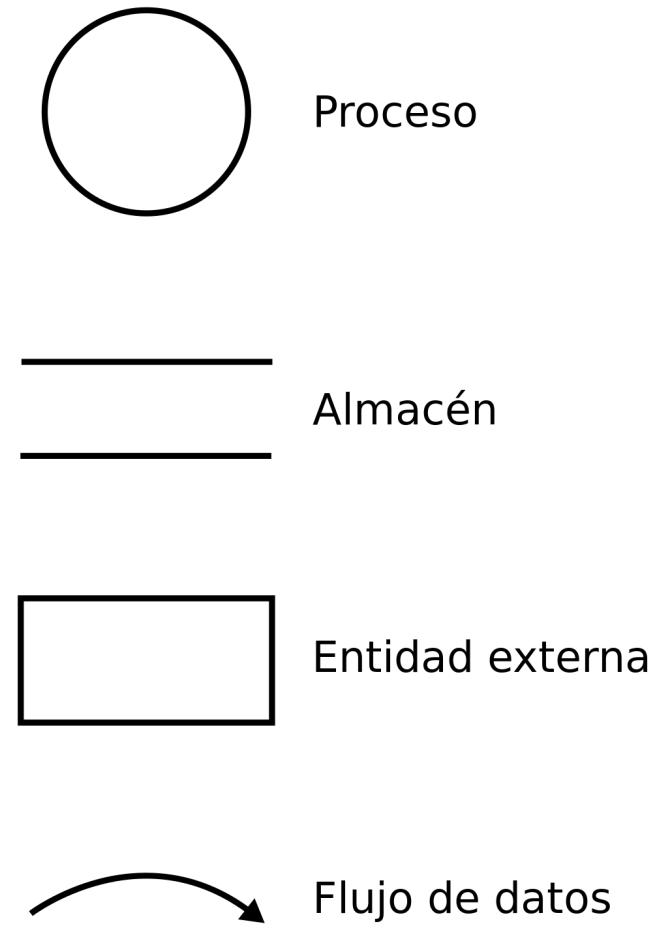
- Se utiliza para modelar las funciones del sistema y el flujo de datos entre ellas
- A finales de los 70, fue formalizado por Tom De Marco; incluye herramientas de apoyo como el Diccionario de Datos y las Especificaciones de Procesos
- Un DFD (diagrama de flujo de datos) es una representación gráfica que muestra cómo la información fluye y se transforma desde la entrada hasta la salida del sistema
- Usa niveles de DFD para ir de funciones generales a tareas detalladas, brindando una visión estructurada



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Procesos

- Los procesos transforman los datos de entrada en salidas, representando funciones del sistema, no programas en ejecución
- Regla de conservación de datos: Las salidas deben generarse a partir de las entradas y cualquier dato adicional. Si faltan o sobran datos, se considera un error o pérdida de información
- Normas de nomenclatura:
  - Numeración y nombre único en todo el DFD
  - Nombre breve y preciso que describa la función
  - No incluye detalles técnicos, solo la lógica del sistema



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Almacenes de Datos

- Los almacenes de datos representan información almacenada temporalmente, conocida como “datos en reposo”. Funcionan como dispositivos lógicos de almacenamiento y pueden contener cualquier tipo de datos, sin importar la tecnología utilizada
- Características:
  - Legibilidad: Pueden repetirse en un DFD para mayor claridad
  - Ubicación en Niveles: Aparecen en el nivel más alto donde conectan varios procesos y en niveles inferiores solo cuando interactúan directamente con procesos específicos
  - Almacenes Locales: Si solo se conecta con un proceso, se representa en el nivel correspondiente



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Almacenes de Datos

- Tipos de Estructuras:
  - Estructura Simple: Compuesta por atributos que identifican una ocurrencia
  - Estructura Compleja: Representada mediante un Diagrama Entidad-Relación (ER) para mostrar su organización interna
- La información de cada almacén se documenta en el Diccionario de Datos para garantizar precisión y consistencia



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Entidades Externas

- Las entidades externas son elementos que interactúan con el sistema sin formar parte de él, como personas, departamentos o subsistemas. Representan las fuentes que envían o reciben información del sistema
- Características:
  - Establecen la interfaz entre el sistema y su entorno
  - Deben tener un nombre claro que describa su función o relación con el sistema
  - Pueden aparecer varias veces en un DFD para mejorar la claridad
- Estas entidades se representan en el Diagrama de Contexto (nivel más alto) para mostrar la interacción del sistema con el entorno, aunque también pueden aparecer en diagramas de niveles inferiores



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- Un flujo de datos es el camino por donde circulan los datos entre partes del sistema, representado por arcos dirigidos.
- Se dividen en:
  - Flujos de Datos Discretos: Transportan información en un momento específico, como una solicitud de datos puntual
  - Flujos de Datos Continuos: Son flujos discretos que permanecen activos a lo largo del tiempo, como en procesos que monitorean continuamente el estado de un sistema



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- Se muestran dos tipos de conexiones entre las entidades externas y los almacenes. El asterisco (\*) indica que esta conexión solo es permitida cuando el almacén de datos externo actúa como interfaz entre el sistema y la entidad externa, y solo aparece en el diagrama de contexto.

Destino/Fuente	Proceso	Almacén	Entidad Externa
Proceso	SI	SI	SI
Almacén	SI	NO	NO*
Entidad Externa	SI	NO*	NO



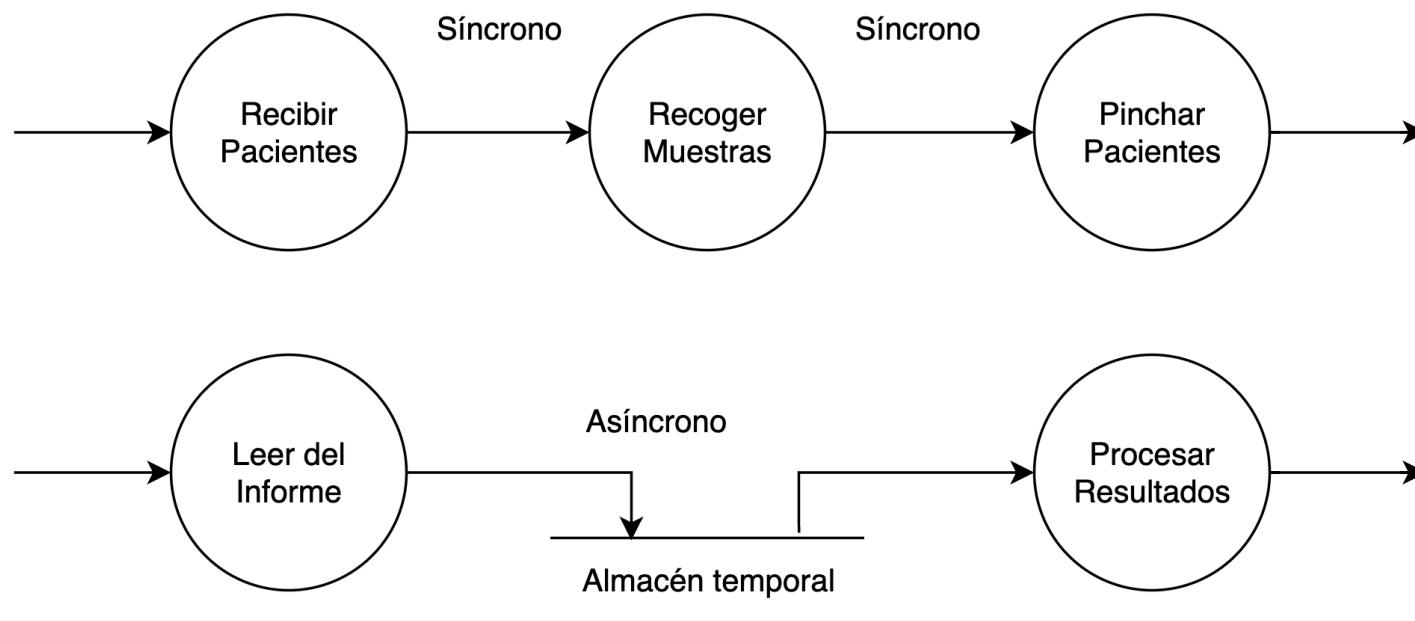
# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- La conexión directa entre dos procesos mediante un flujo de datos es posible si la información es síncrona, es decir, si el proceso receptor comienza su actividad justo cuando el proceso emisor ha completado la suya



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- Las conexiones entre procesos y almacenes son:
  - **Flujo de Consulta:** Permite al proceso acceder al almacén para consultar datos o verificar criterios
  - **Flujo de Actualización:** Modifica el almacén mediante la creación, eliminación o actualización de datos
  - **Flujo de Diálogo:** Combina consulta y actualización, permitiendo acceder y modificar información en un solo proceso



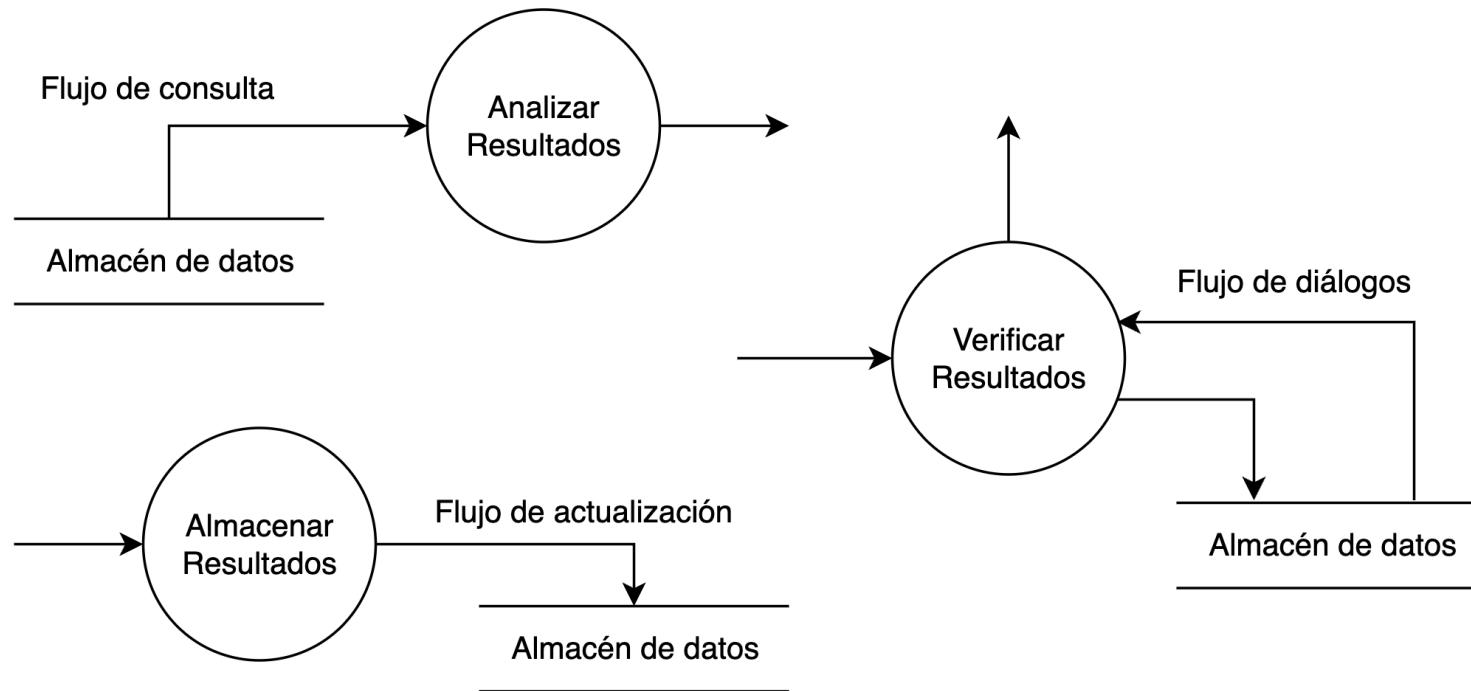
# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- Conexiones entre procesos y almacenes



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Flujo de Datos

- Cada flujo de datos debe tener un nombre que describa claramente la información que transporta. No es necesario nombrar los flujos que ingresan o salen de almacenes simples, ya que su estructura coincide con la del almacén
- Los flujos de datos no indican el control de los procesos ni cuándo deben comenzar o terminar



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Niveles de Abstracción en los DFD

- En los DFD de sistemas grandes, se recomienda estructurarlos en niveles de abstracción de arriba hacia abajo (top-down), donde cada nivel ofrece más detalle que el anterior
- Un DFD grande se organiza jerárquicamente, con cada nivel expandiendo el anterior:
  - Diagrama de Contexto (Nivel 0): Es el más general, define los límites del sistema e identifica las interfaces con el entorno
  - Diagramas de Niveles: Detallan las funciones principales del sistema, que pueden descomponerse en niveles inferiores
  - Procesos Primitivos: Representan funciones básicas que no se descomponen más, describiéndolas con precisión como bloques fundamentales del sistema



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Niveles de Abstracción en los DFD

- Los niveles de descomposición se organizan de la siguiente manera:
  - Nivel 0: Diagrama de Contexto, que define los límites del sistema
  - Nivel 1: Diagrama de Subsistemas, mostrando las principales divisiones funcionales
  - Nivel 2: Diagramas de Funciones de Subsistemas, detallando las operaciones dentro de cada subsistema
  - Nivel N: Diagramas de Procesos, describiendo subfunciones en detalle según sea necesario



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Niveles de Abstracción en los DFD

- Para garantizar consistencia, los flujos de datos en cada nivel deben coincidir con los del nivel inferior, según la [regla de balanceo](#), que exige que todos los flujos de datos de un diagrama hijo estén representados en el diagrama padre. Las salidas también deben coincidir, salvo en casos excepcionales.
- La numeración de los DFD sigue un esquema estructurado:
  - El Diagrama de Contexto se numera como 0
  - Los Diagramas de Nivel 1 del 1 al N
  - Los niveles siguientes usan un sistema de numeración anidada, similar al sistema Dewey, para facilitar la referencia y coherencia

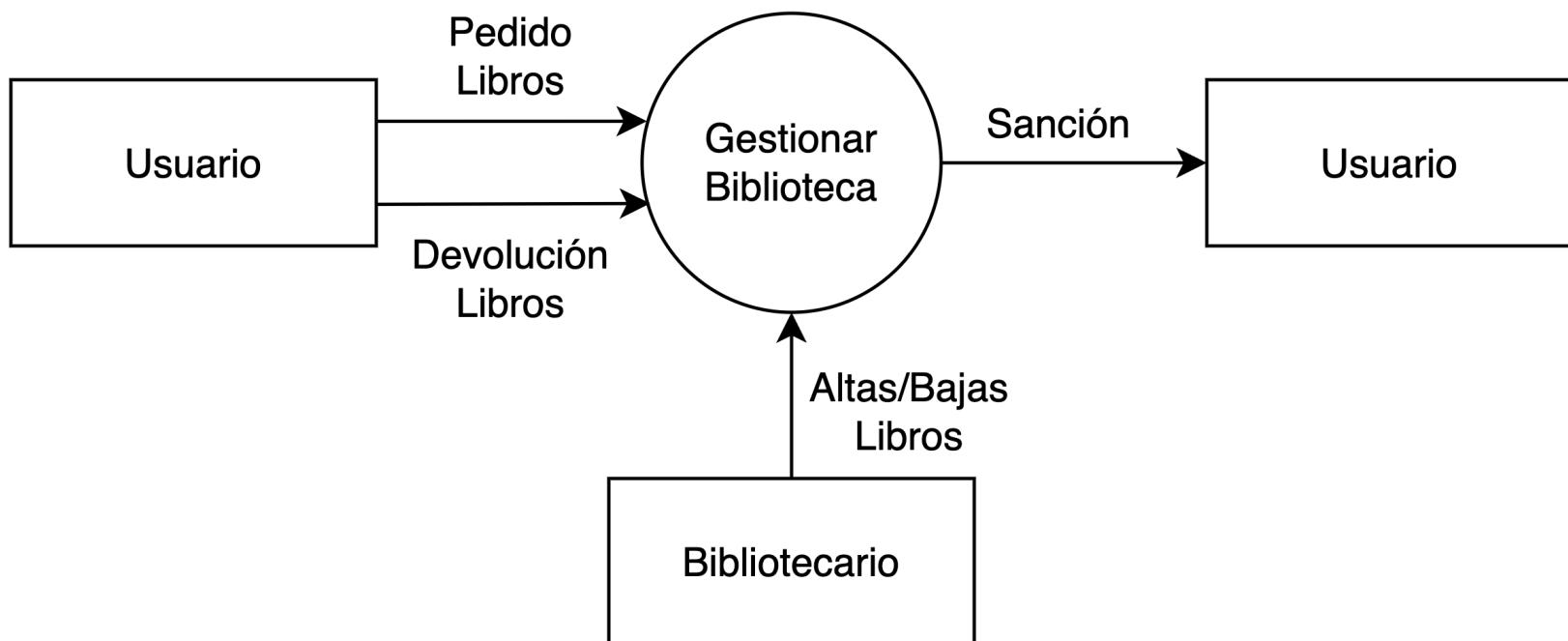


# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

#### Diagrama de Contexto

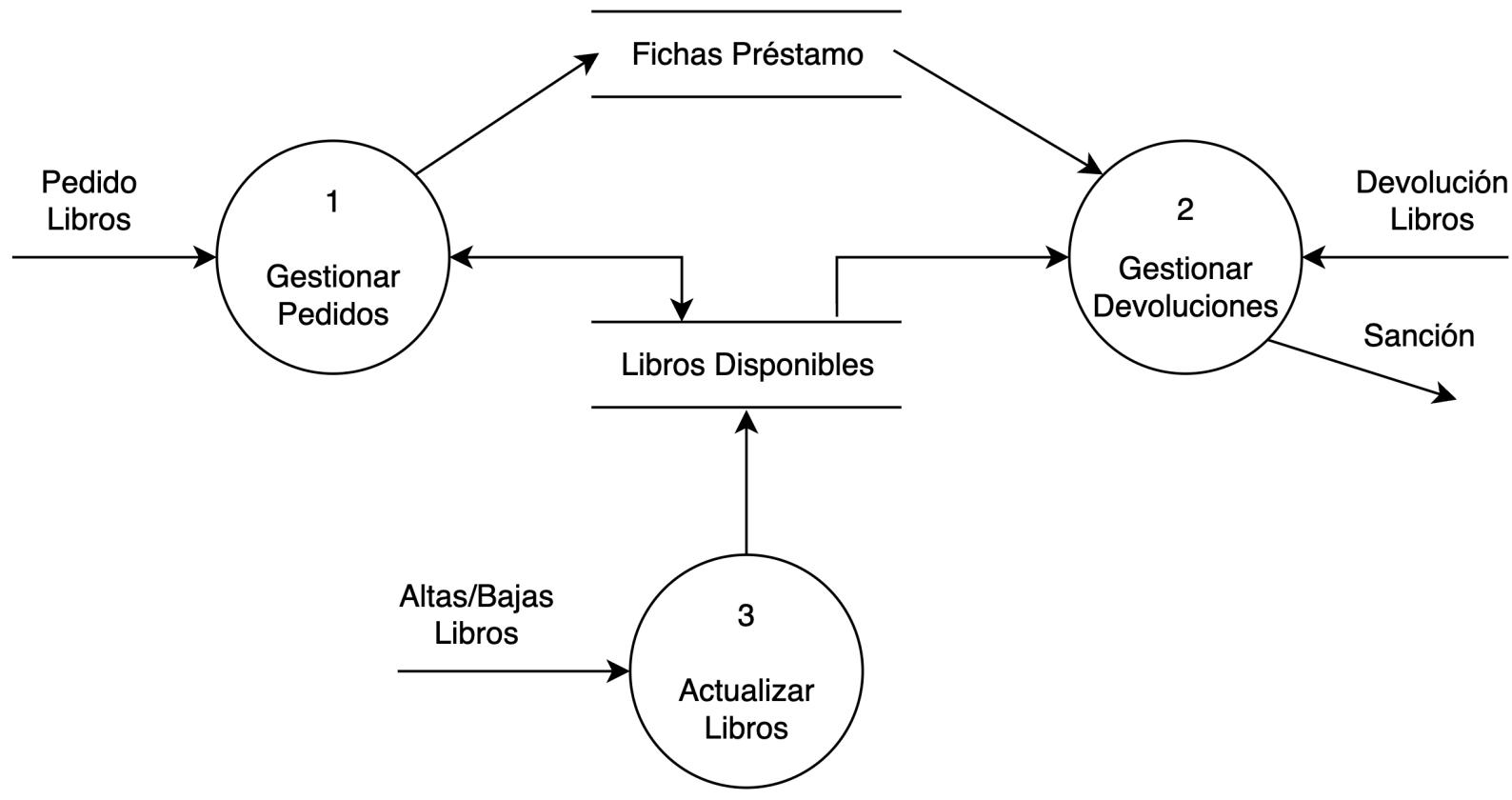


# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

Diagrama 1: Gestionar Biblioteca



# Técnicas Estructuradas

## Clasificación según el Enfoque de Modelado

### Diagrama de Flujo de Datos

Diagrama 2: Gestionar Devoluciones

