

MODELADO ORIENTADO A OBJETOS CON UML

Tema 4

Ingeniería del Software
ETS Ingeniería Informática
DSIC – UPV

Curso 2024-2025

Objetivos

- Mostrar la necesidad de construir modelos para resolver problemas complejos y de grandes dimensiones
- Comprender qué es el modelado conceptual y distinguirlo claramente del diseño
- Aprender un subconjunto de UML, como notación de modelado OO
- Modelado estructural de un sistema
- Modelado del comportamiento de un sistema

Contenidos

1. Motivación.

2. Modelado OO


- Visión de un sistema software OO

3. Notación UML

- Diagrama de Clases. **Parte 1**
- Diagrama de Casos de Uso. **Parte 2**
- Diagramas de Secuencia
- Otros diagramas

Bibliografía básica

 Booch, G., Rumbaugh, J., Jacobson, I., UML. El Lenguaje Unificado de Modelado. UML 2.0 2ª Edición. Addison-Wesley, 2006

 Stevens, P., Pooley, R. Utilización de UML en Ingeniería del Software con Objetos y Componentes. 2ª Edición. Addison-Wesley Iberoamericana 2007Ingeniería del Software. (8ª ed.). Addison-Wesley, 2008

 Weitzenfeld, A., Ingeniería del Software OO con UML. Java e Internet. Thomson, 2005

 <https://www.uml.org/>

Motivación

¿Qué es un modelo?

“Un modelo es una simplificación de la realidad”

¿Por qué modelamos?

Construimos modelos para comprender mejor el sistema que estamos desarrollando

- Nos ayudan a **visualizar** cómo es o queremos que sea un sistema.
- Nos permiten **especificar** la estructura o el comportamiento de un sistema
- Nos proporcionan plantillas que nos guían en la **construcción** de un sistema
- **Documentan** las decisiones que hemos adoptado

Motivación

Modelado Orientado a Objetos

- Aparecen los lenguajes de programación OO
- Se requiere un nuevo enfoque de análisis y diseño

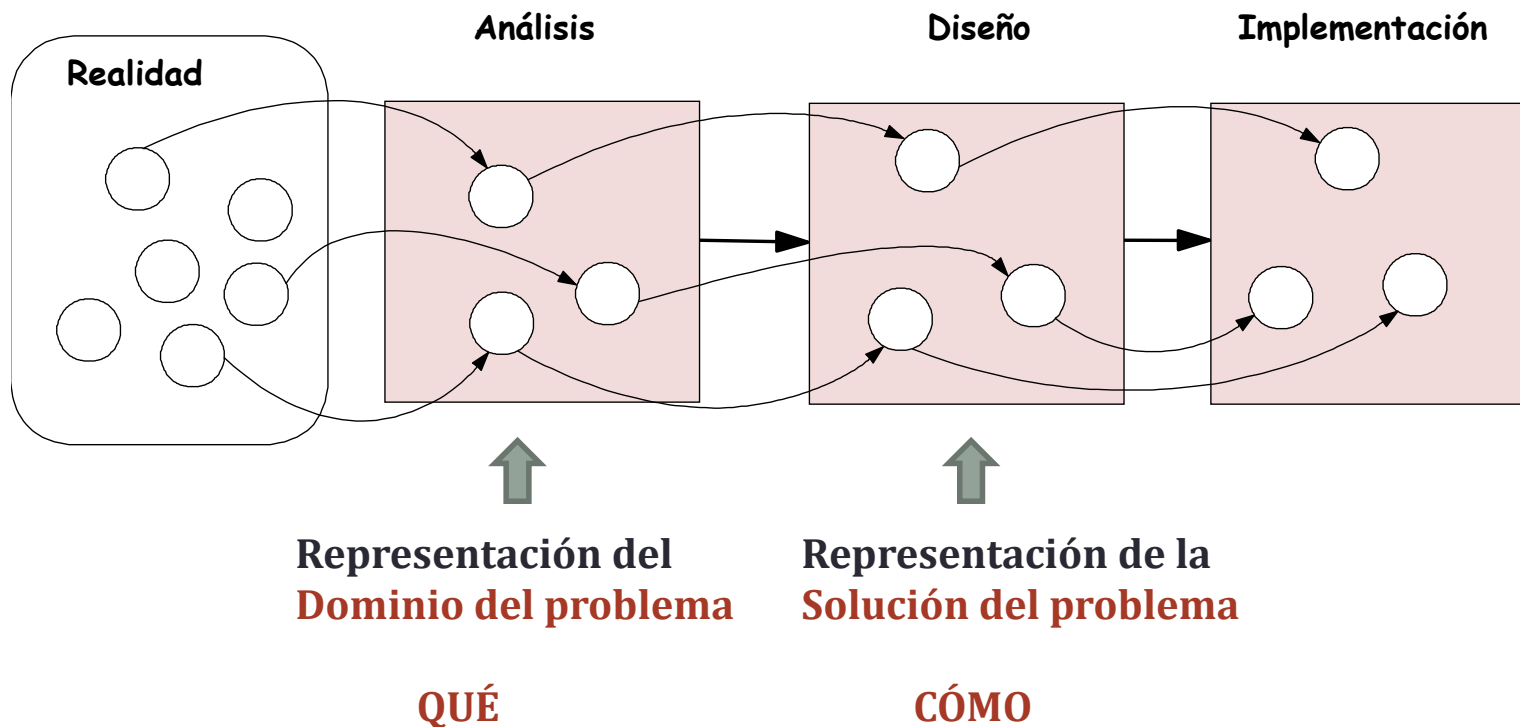


*“Un proceso que examina los requisitos desde la perspectiva de las **clases** y **objetos** encontrados en el vocabulario del dominio del problema” (Booch, 1994)*

1. Próximo a los **mecanismos cognitivos humanos**
2. Desarrollo **incremental** bajo una **noción** común de **objeto**

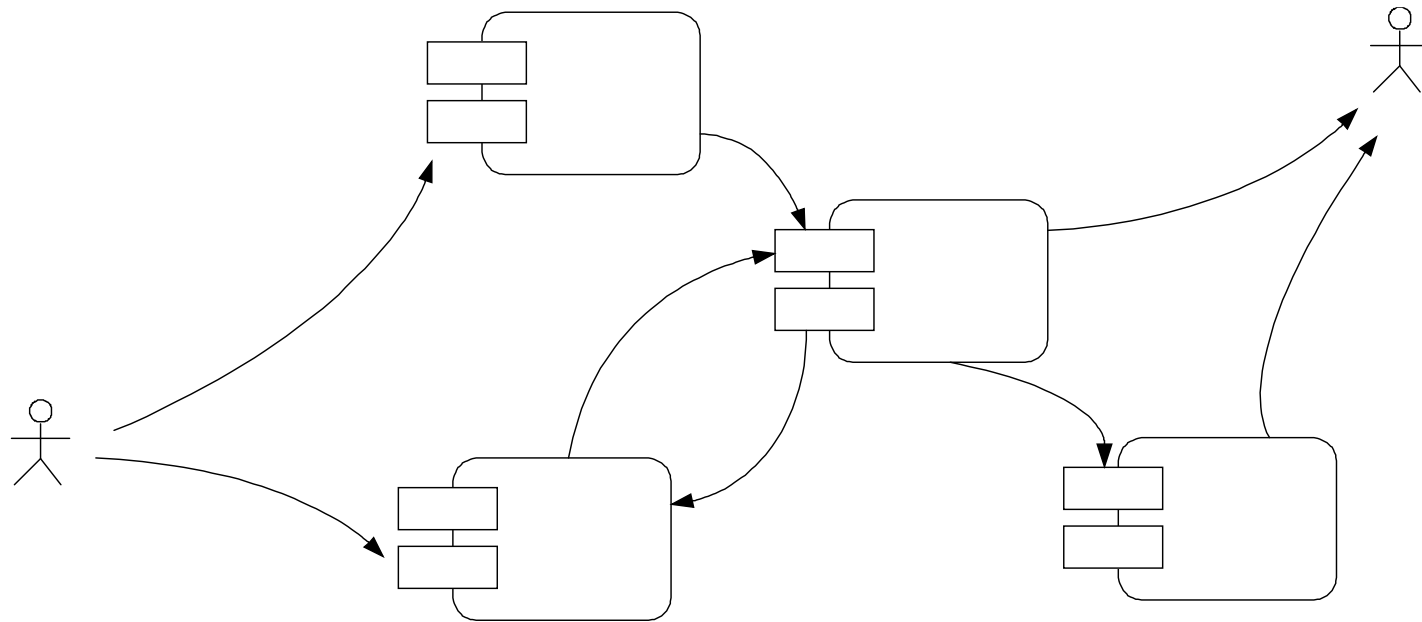
Motivación

En el enfoque OO, la descomposición del sistema se basa en los *objetos* o *clases de objeto* que se descubren en el dominio del problema



Visión de un Sistema Software OO

Visión estática + Visión dinámica



Sistema Software OO - Visión estática

Objeto:

- Entidad que existe en el mundo real
- Tiene identidad, una estructura y un estado

El avión con matrícula 1234
El avión con matrícula 6754

Clase:

- Describe un conjunto de objetos con las mismas propiedades y un comportamiento común.
- Relaciones entre clases

Avión

Sistema Software OO - Visión dinámica

- Los objetos se comunican mediante la invocación de métodos de otros objetos.
- Se describen aspectos de un sistema que cambian con el tiempo.
 - Interacciones entre objetos
 - Posibles estados de un objeto
 - Transiciones entre estados
 - Qué eventos se producen
 - Qué operaciones se ejecutan

UML



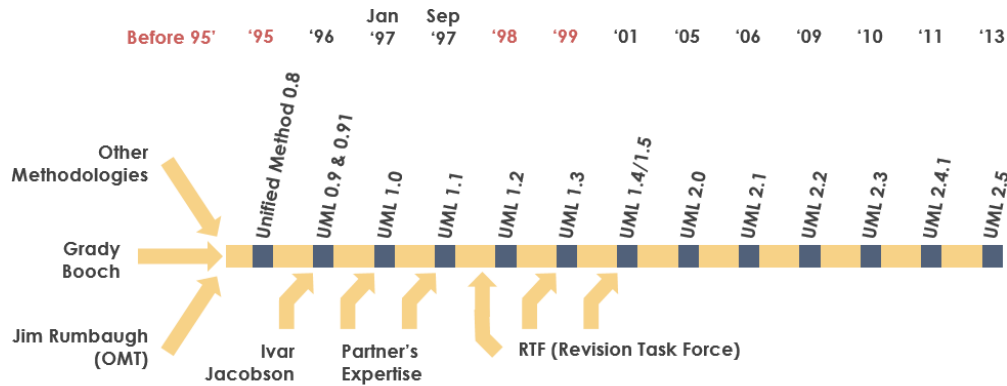
Unified Modeling Language



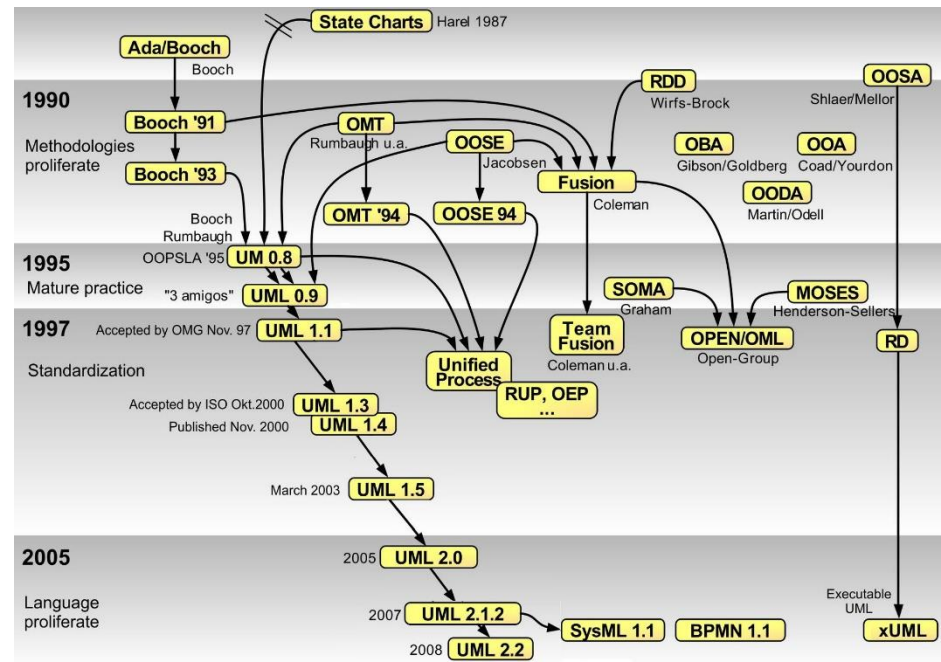
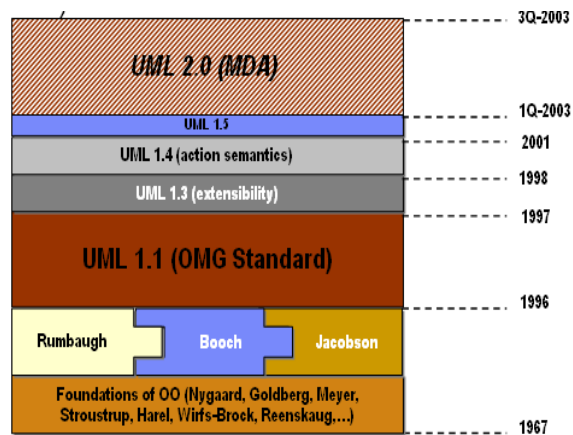
The Unified Modeling Language™ (UML™) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, creating a "blueprint" for construction.

- Visualizar
- Especificar
- Construir
- Documentar

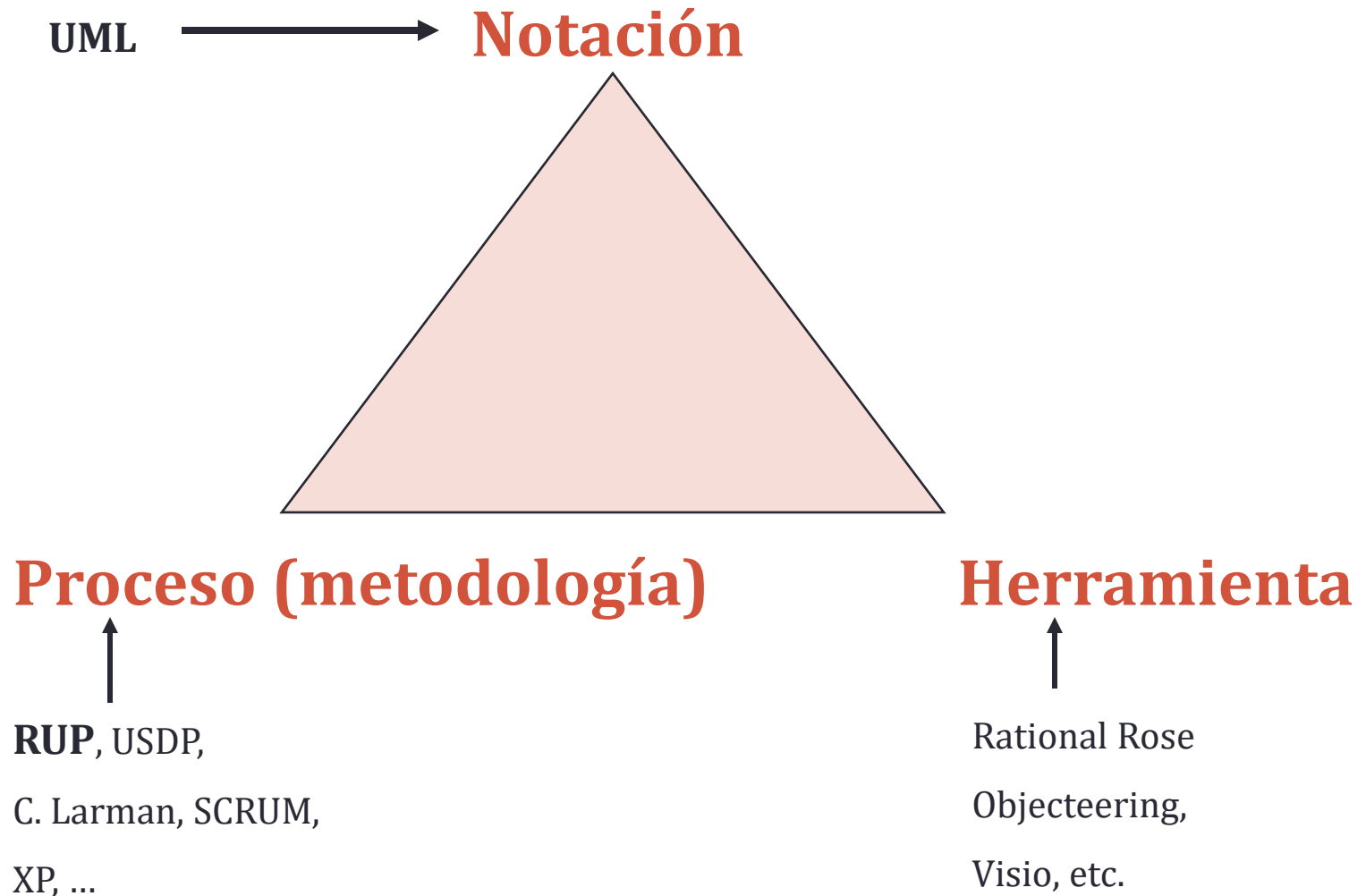
UML - Evolución



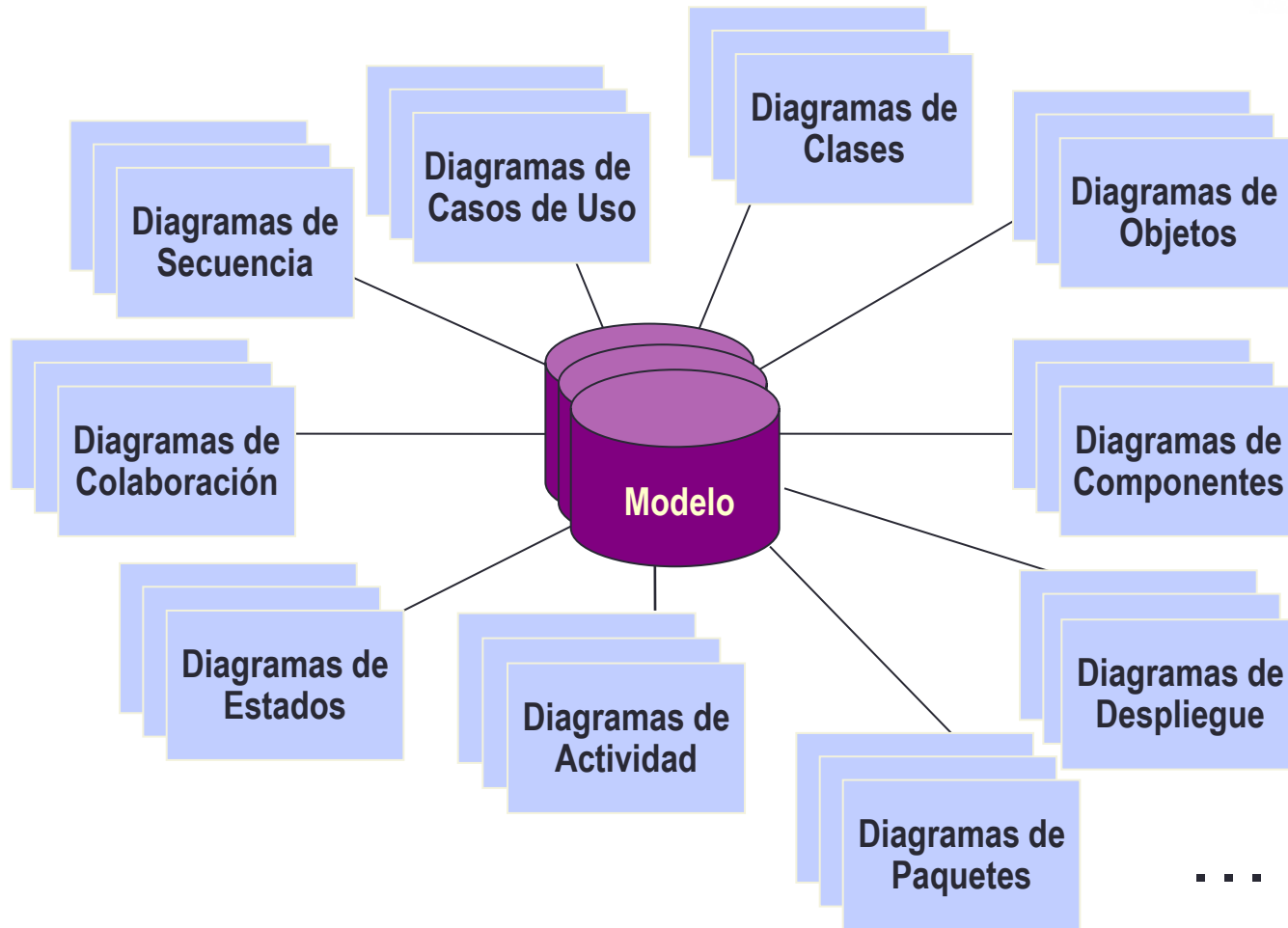
Before '95 - Fragmentation ▶ '95 - Unification ▶ '98 - Standardization ▶ '99 - Industrialization



UML - Triángulo del éxito



UML



UML - Tipos de diagramas

Diagrama de Clase (incluyendo Diagrama de Objetos). **Parte 1**

Diagrama de Casos de Uso. **Parte 2**

Diagramas de Comportamiento

Diagrama de Estados

Diagrama de Actividad

Diagramas de Interacción

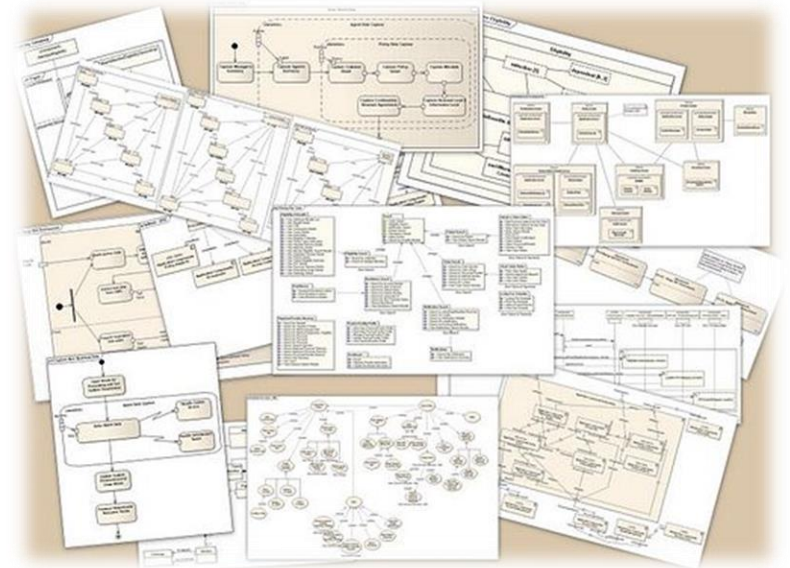
Diagrama de Secuencia

Diagrama de Colaboración

Diagramas de implementación

Diagrama de Componentes

Diagrama de Despliegue





Parte 1

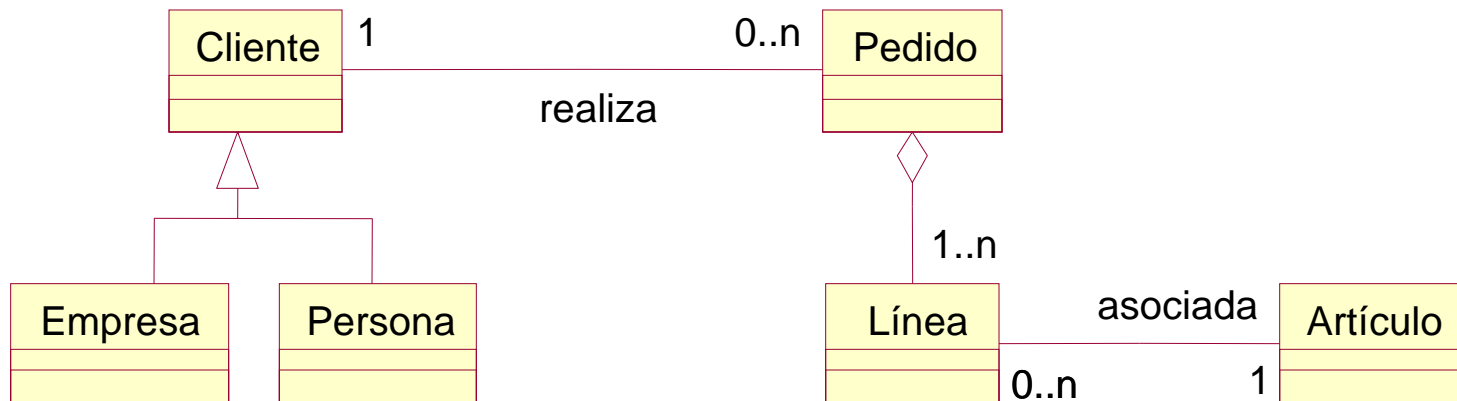
Diagrama de Clases

- Clases – Objetos
- Relaciones entre clases
 - Asociación
 - Agregación
 - Composición
 - Especialización/
Generalización
(Herencia)

Diagrama de Clases

- Muestra la estructura estática del sistema, mostrando las clases y las relaciones entre ellas
- Es la herramienta principal de la mayor parte de los métodos OO

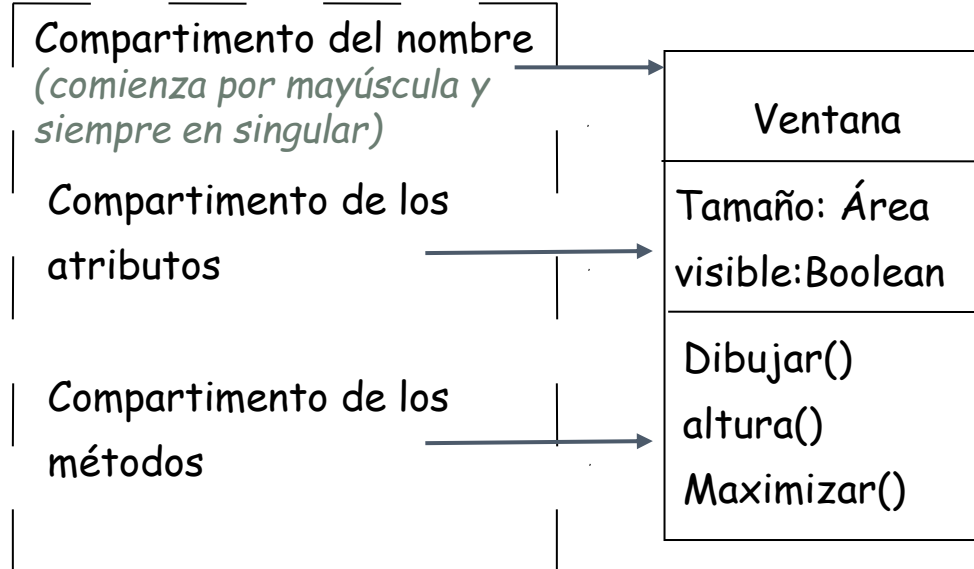
Notación



Clase

Es la descripción de un grupo de objetos con estructura, comportamiento y relaciones similares

Notación



Clase

- Los atributos/operaciones pueden ser:

- (-) Privados
- (#) Protegidos
- (+) Públicos

Reglas de visibilidad

+ Atributo público : int
Atributo protegido : int
- Atributo privado : int

+ "Operación pública"
"Operación protegida"
- "Operación privada"

- Los atributos se pueden representar mostrando únicamente el nombre
- Los atributos no incluyen referencia a otros objetos, estas referencias se representan mediante enlaces
- Un **atributo derivado** se representa como /Atributo : Tipo
- Un método es la implementación de una operación

Clases / Objetos

Objetos

Clases

Un Arbol Binario:
Arbol Binario

| |
|---|
| <u>Houston: Ciudad</u> |
| Nombre Ciudad: Houston TX poblacion: 3.000.000 |

| |
|-----------|
| (Persona) |
| Pepe |

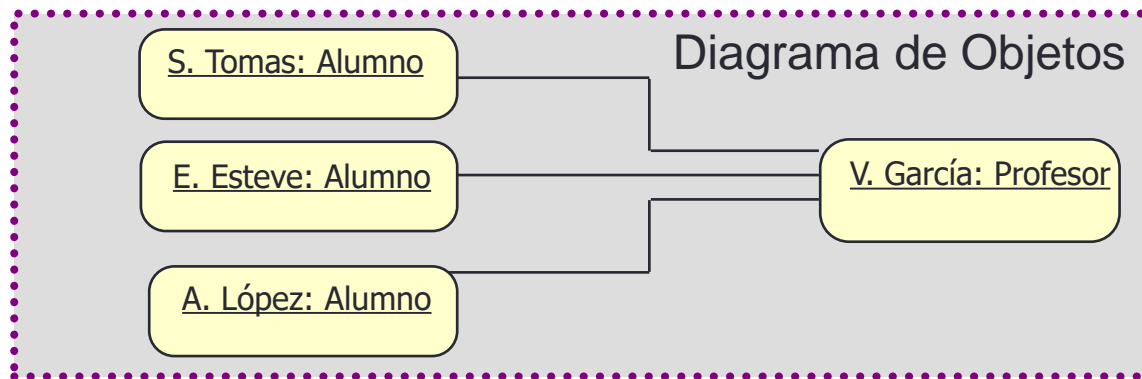
Arbol Binario

| |
|---|
| Ciudad |
| - Nombre Ciudad: String - poblacion:Real |

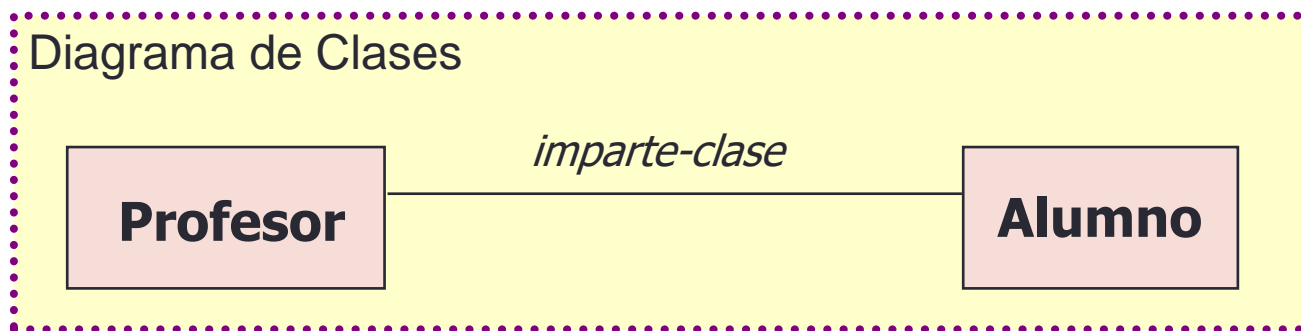
| |
|----------------|
| Persona |
| Nombre:String |

Asociaciones

- Un enlace es una conexión física o conceptual entre objetos
- Una asociación es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro



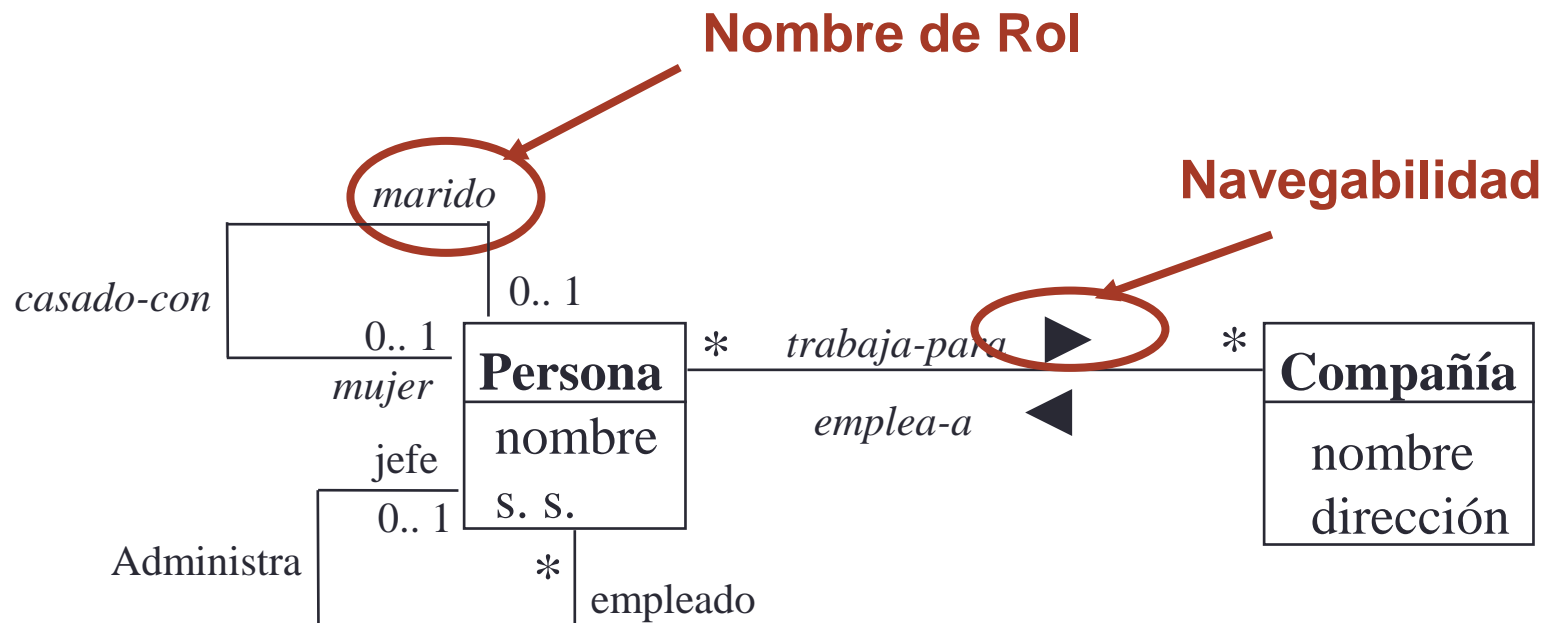
Enlace



Asociación

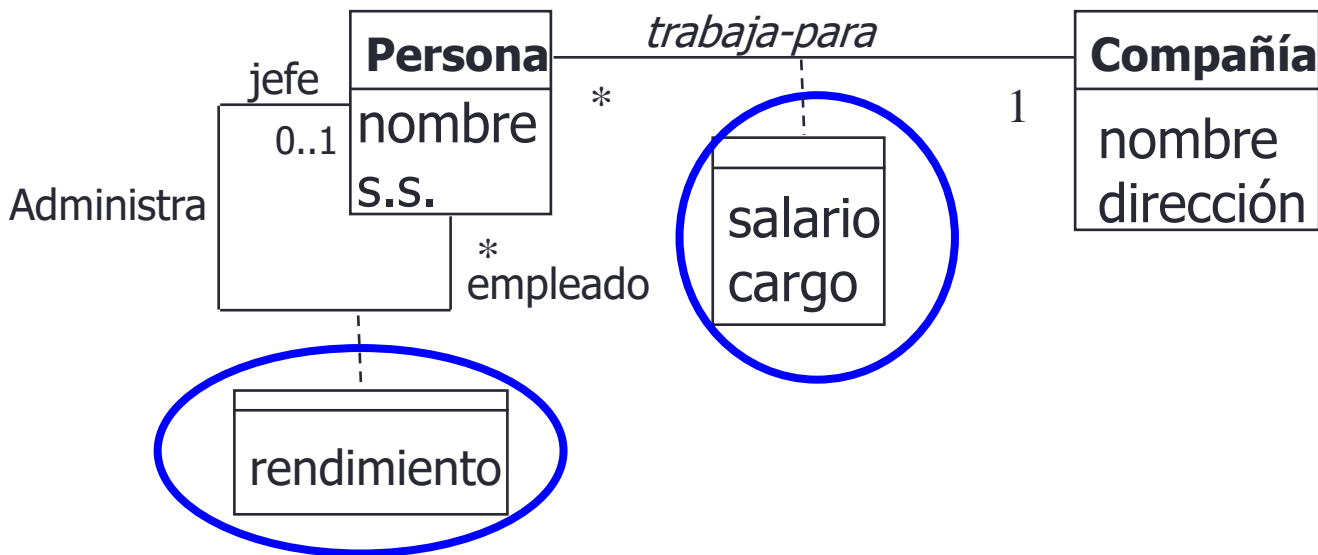
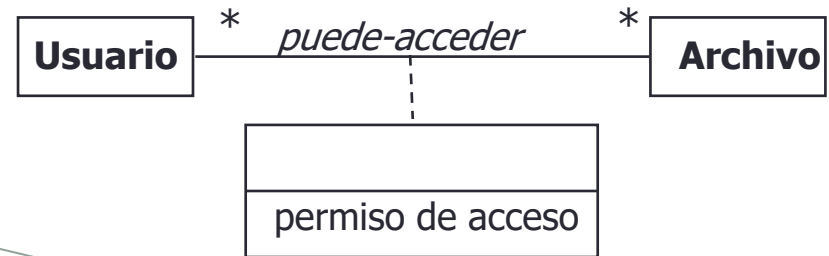
Asociaciones

- Toda asociación es bidireccional, es decir, se puede navegar en los dos sentidos, desde objetos de una clase a objetos de la otra.
- Tiene un nombre
- Puede tener nombres de rol en los extremos (obligatorio en asociaciones reflexivas)
- Multiplicidad: 1, 0..1, 0..N (*), 1..N, M..N



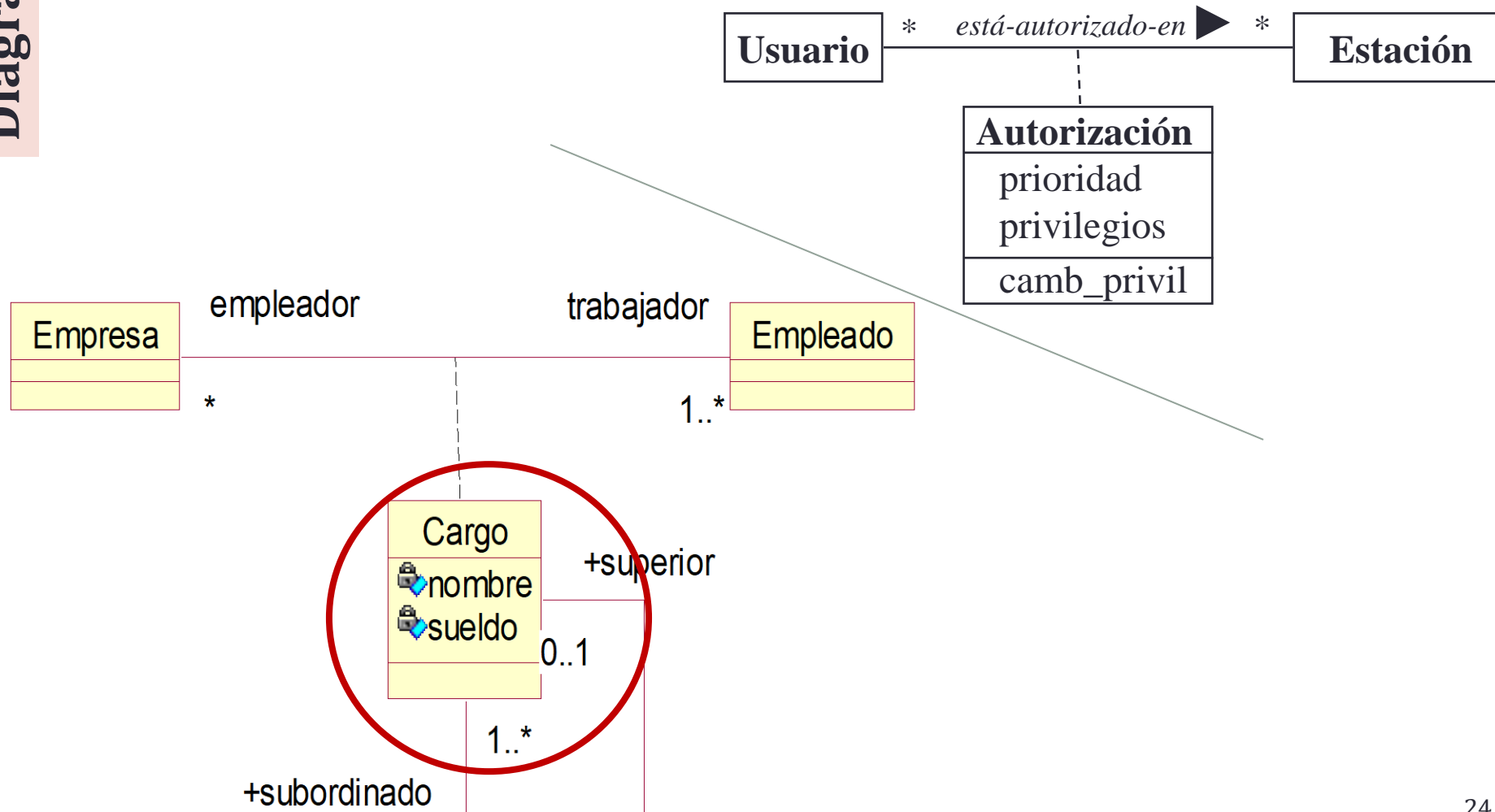
Asociaciones - Atributos de Enlace

En una asociación entre clases, la propia relación puede tener propiedades, denominados atributos de enlace



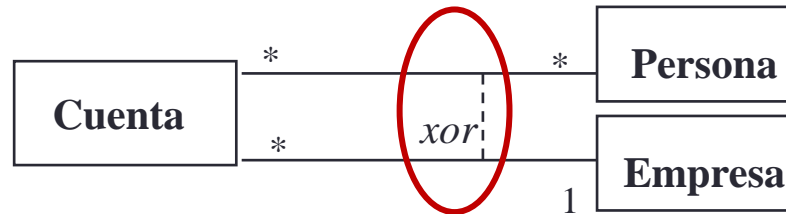
Asociaciones - Clase Asociación

El atributo de enlace puede ser una clase



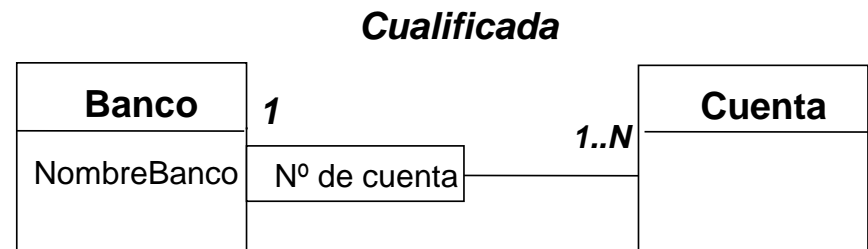
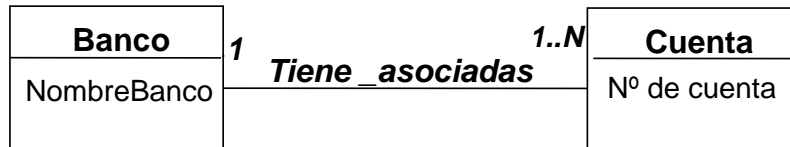
Asociaciones - Asociación excluyente

XOR



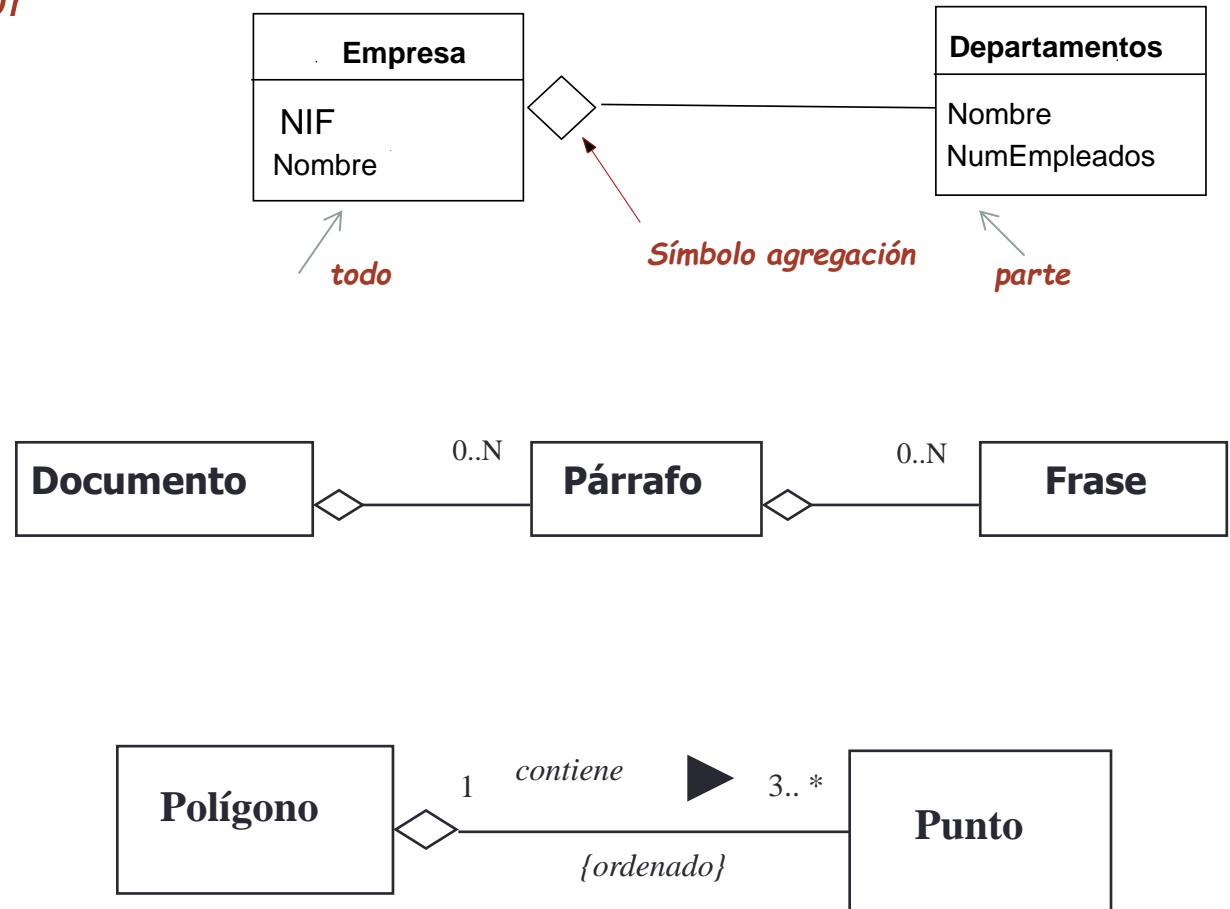
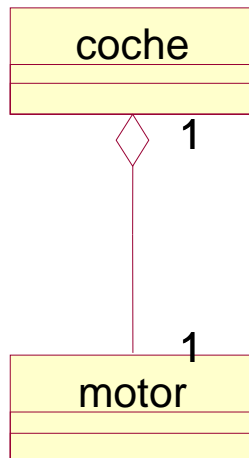
Asociaciones - Asociación cualificadas

Cualificadores o calificadores nos sirven para refinar más el modelo, indicando el índice para recorrer la relación (*¿Cómo identificar un objeto o conjunto de objetos en el otro extremo?*)



Agregación

- Es una asociación con unas propiedades semánticas adicionales.
- “*está formado por*”



Agregación

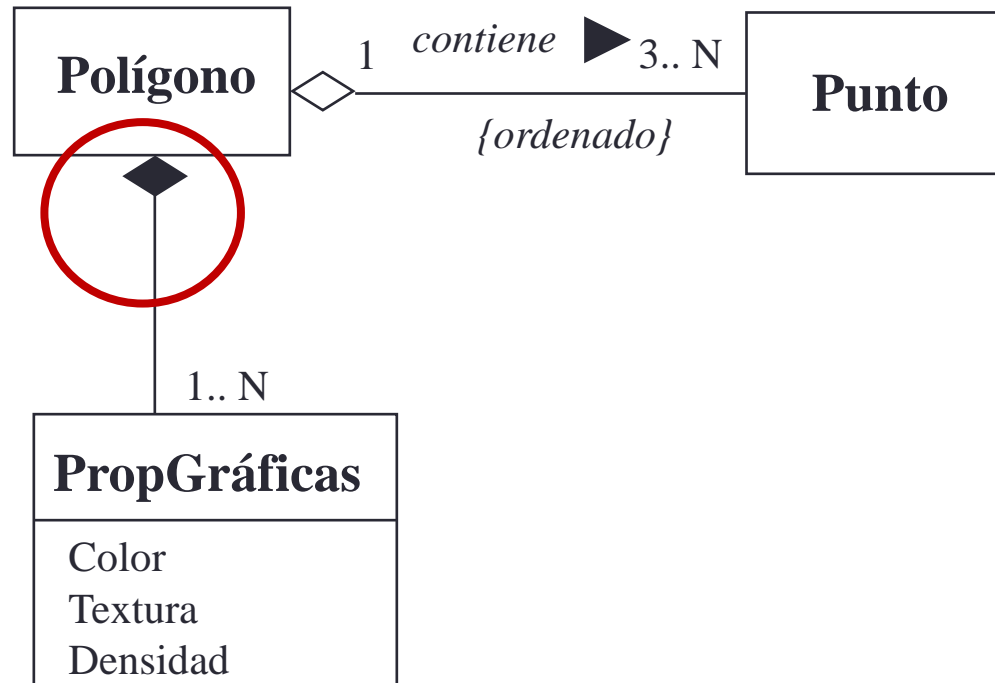
Tipos de agregación:

Inclusiva o física: cada componente puede pertenecer a lo sumo a un compuesto. La destrucción del compuesto implica la destrucción de las partes.

Referencial o de catálogo: los componentes son reutilizables a lo largo de distintos compuestos. No están relacionados los tiempos de vida.

Composición

- Es una agregación inclusiva o física
- “*está compuesta por*”

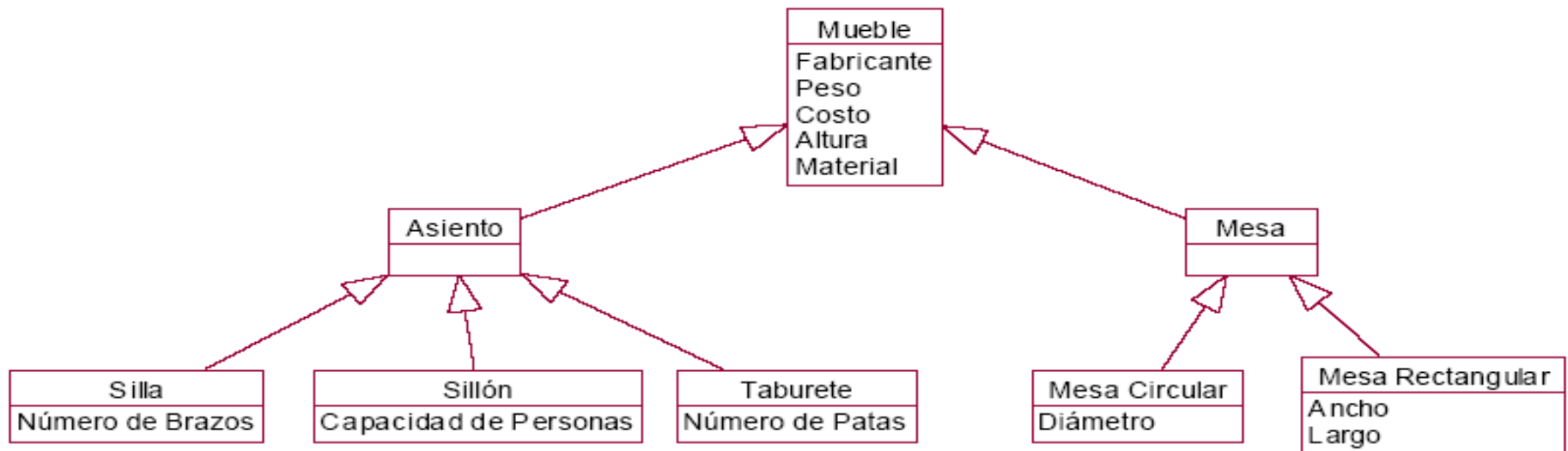


Diferencias entre Composición y Agregación

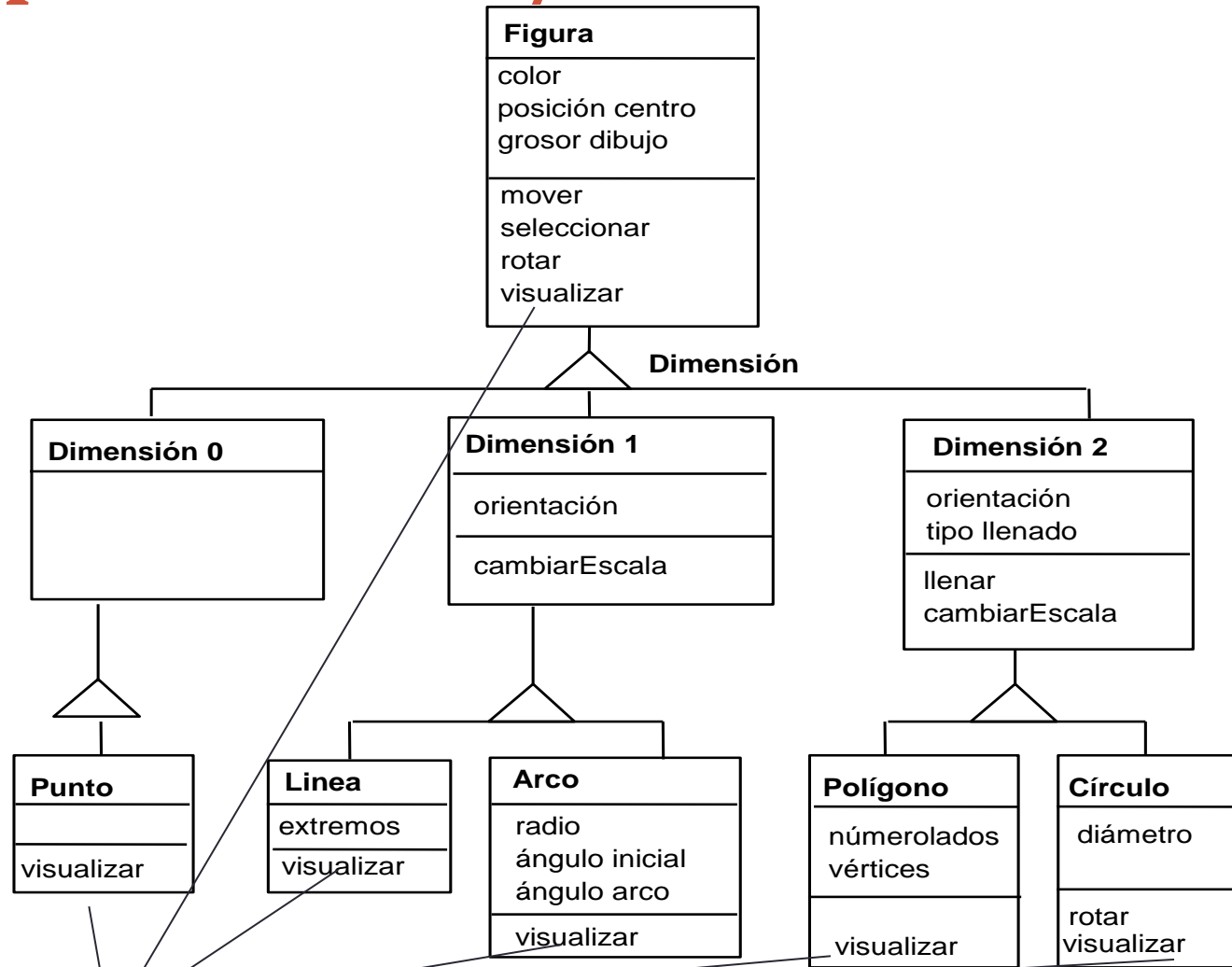
| | Agregación | Composición |
|---|---|---|
| Varias asociaciones comparten los componentes | Sí | No |
| Destrucción de los componentes al destruir el compuesto | No | Sí |
| Cardinalidad a nivel de compuesto | Cualquiera | 0..1 ó 1 |
| Representación |  |  |

Especialización / Generalización

- Permiten definir jerarquías de clases
- Generalización: Dado un conjunto de clases, si tienen en común atributos y métodos, se puede crear una clase más general (superclase) a partir de las iniciales (subclases)
- Especialización: es la relación inversa
- “es un”



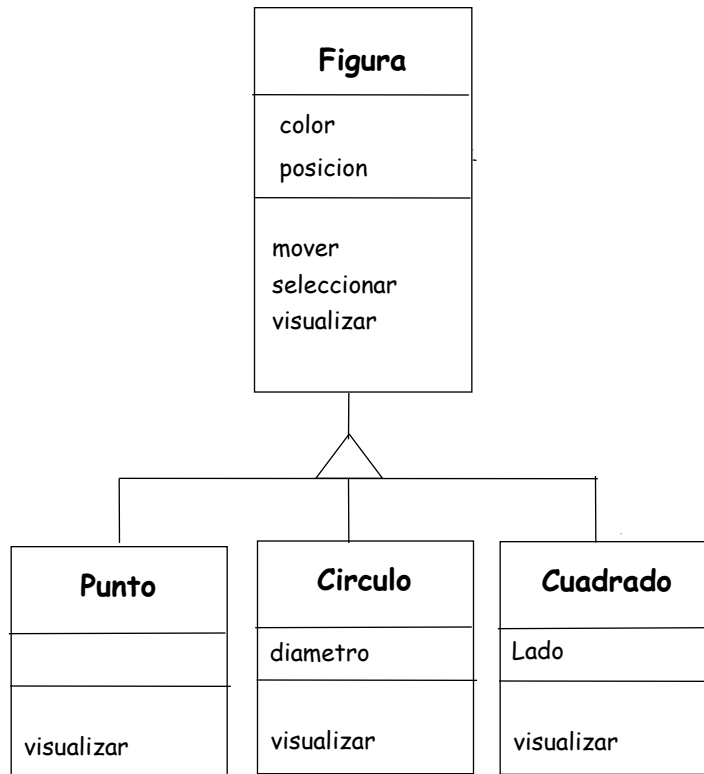
Especialización / Generalización



Cuando en una jerarquía de especialización se repite una característica de una clase (atributo u método) estamos **redefiniendo** la característica heredada

Especialización / Generalización

La relación de **especialización** se emplea en la fase de modelado de un sistema, mientras que la relación de **herencia** se ve como un mecanismo de reutilización de código en la fase de implementación o diseño.

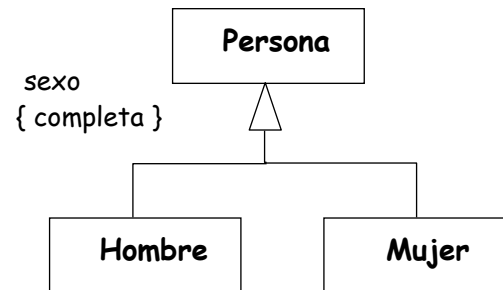


En Implementación, la herencia permite **reutilizar** los atributos y operaciones de la clase **Punto**

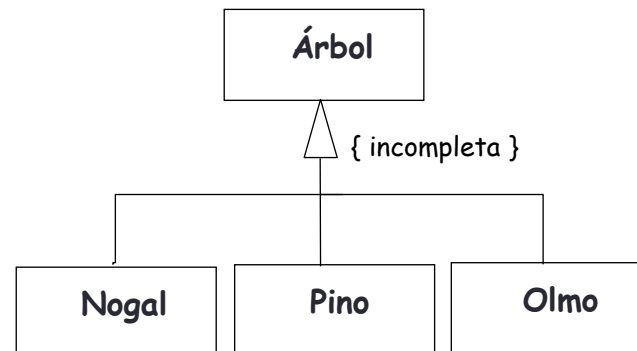
Especialización / Generalización

Dos tipos de **restricciones**:

- **Completa**: Todos los hijos de la generalización se han especificado en el modelo

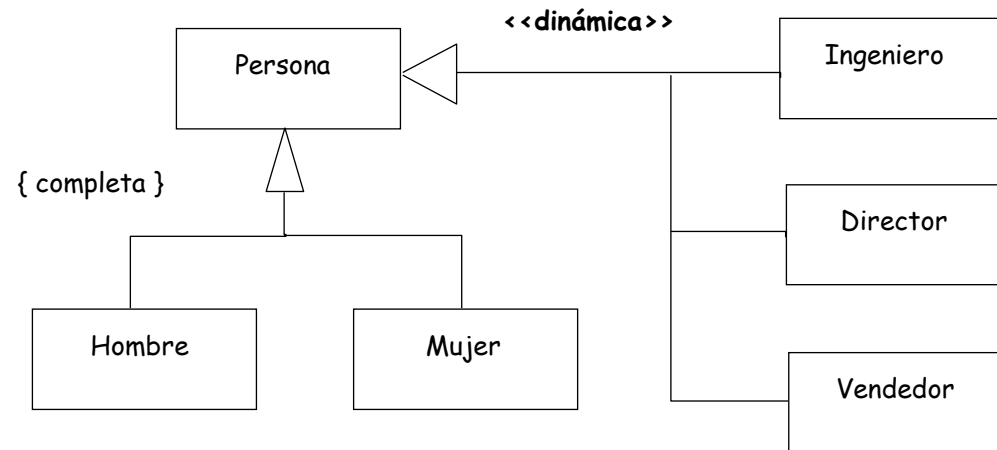


- **Incompleta**: No se han especificado todos los hijos y se permiten hijos adicionales

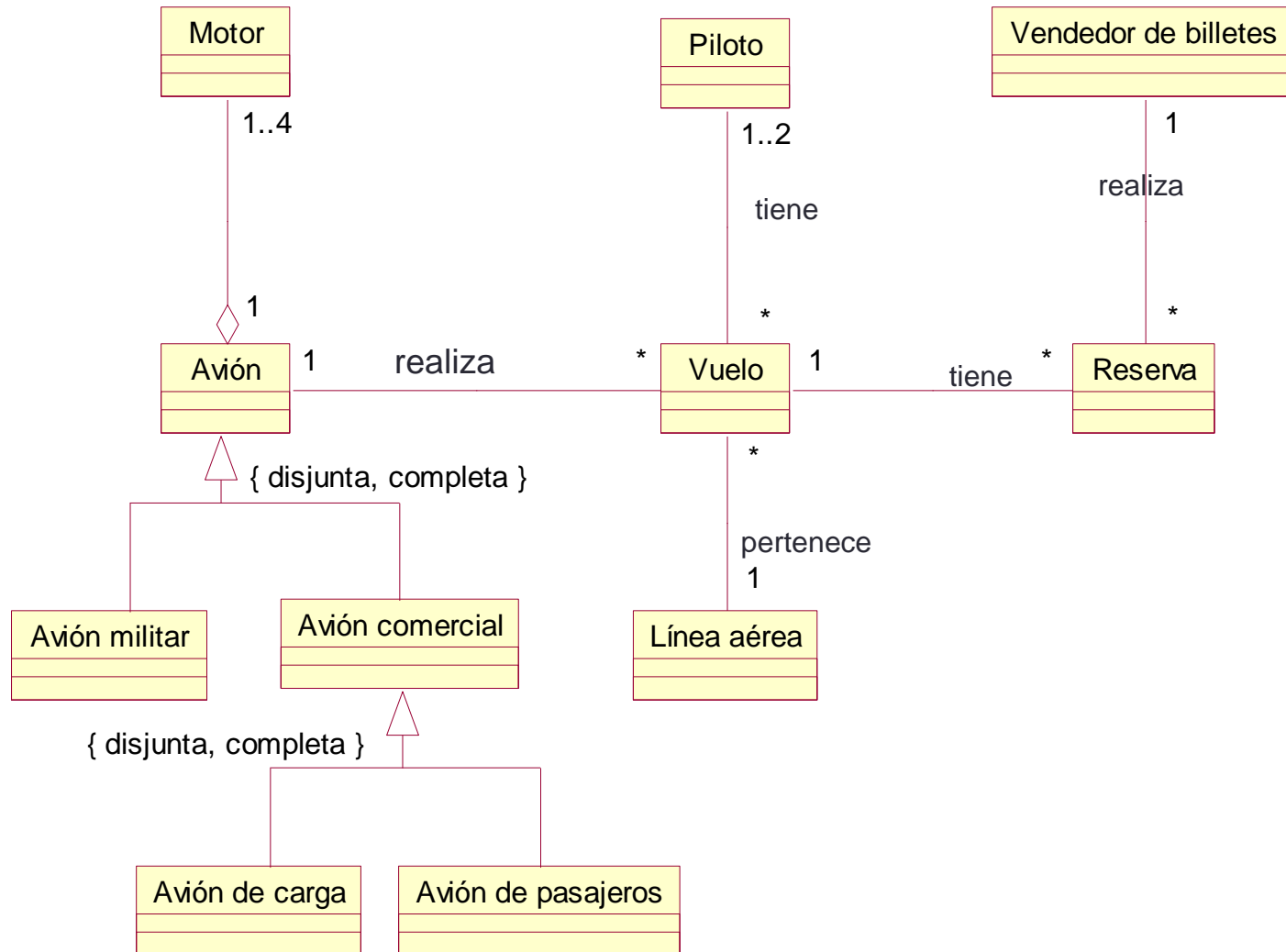


Especialización / Generalización

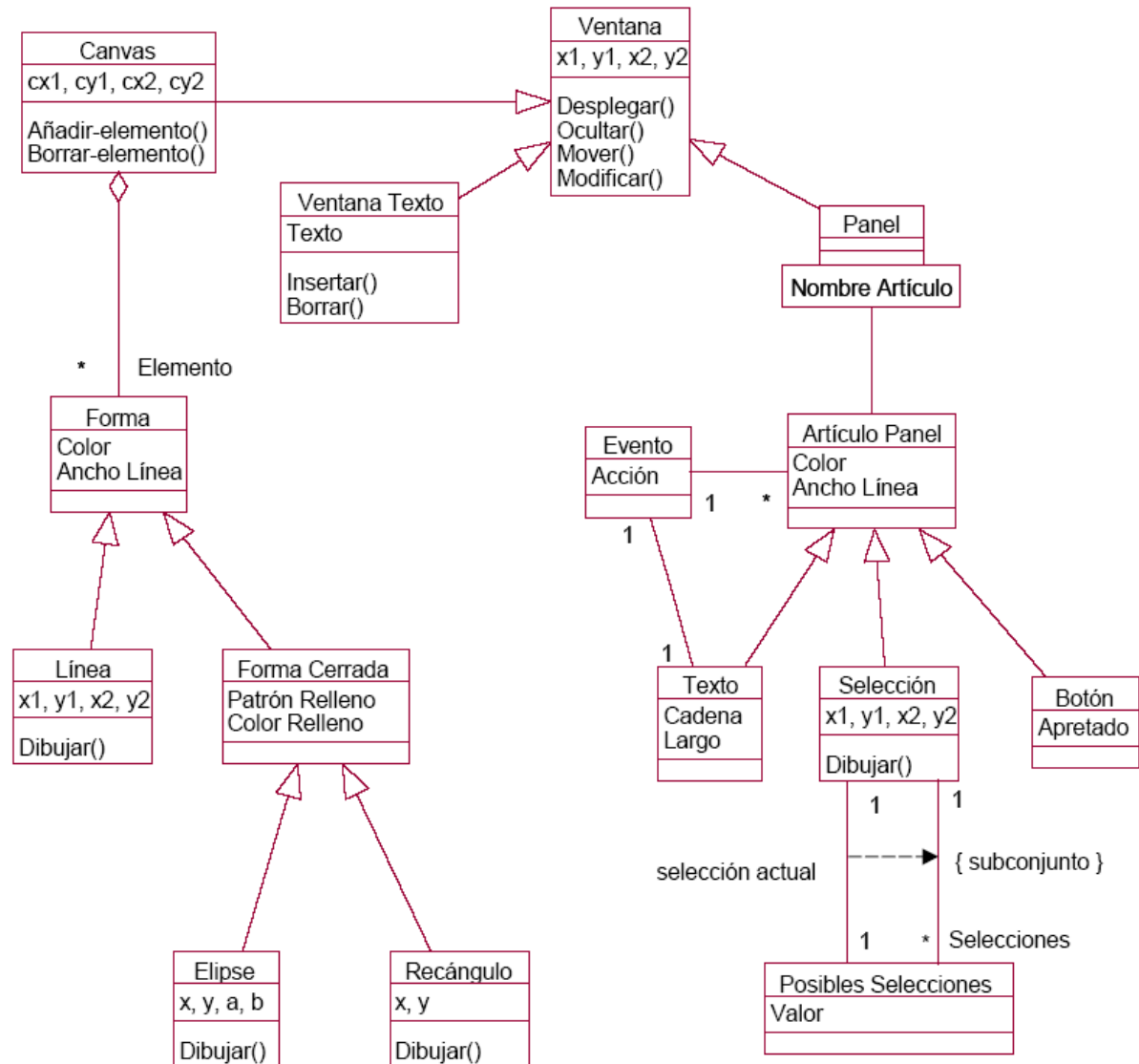
Se habla de especialización **dinámica** cuando un objeto puede cambiar de clase dentro de una jerarquía de subclases



Ejemplos



Ejemplos





Parte 2

Comportamiento

Contenidos

1. Motivación.

2. Modelado OO

- Visión de un sistema software OO

3. Notación UML

- Diagrama de Clases. **Parte 1**
- **Diagrama de Casos de Uso** **Parte 2**
- Diagramas de Secuencia
- Otros diagramas

Modelar el comportamiento de un sistema...

- **Diagramas de Casos de Uso**

- Se utiliza para capturar los requisitos funcionales de la aplicación (sistema) a desarrollar.
- Los diagramas de casos de uso muestran cómo se comunica la aplicación (principales funcionalidades/casos de uso) con el entorno (actores).

- **Diagramas de Interacción**

- Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones.
- Los diagramas de interacción muestran cómo se comunican los objetos para realizar un servicio (muestran la comunicación dentro del sistema).
- Hay dos tipos:
 - **Diagramas de Secuencia** y
 - Diagramas de Colaboración

- Otros diagramas (estados, actividades...)



Parte 2

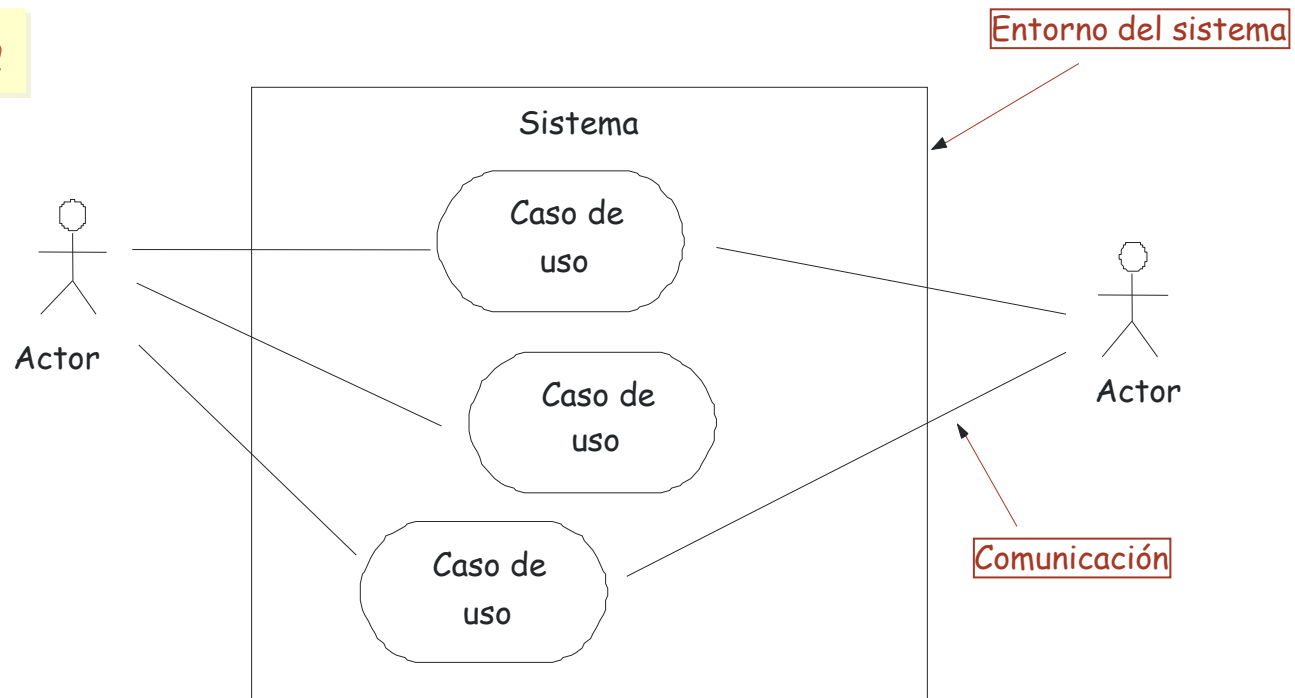
Casos de Uso

- Actores y Casos de Uso
- Relaciones
- Diagramas de Casos de Uso
 - Diagrama de contexto y diagrama inicial
 - Plantillas de especificación
 - Proceso de construcción

Casos de Uso

- Es una técnica para capturar información de cómo un sistema trabaja actualmente, o de cómo se desea que trabaje
- Se utiliza para capturar los requisitos funcionales del sistema a desarrollar

Notación

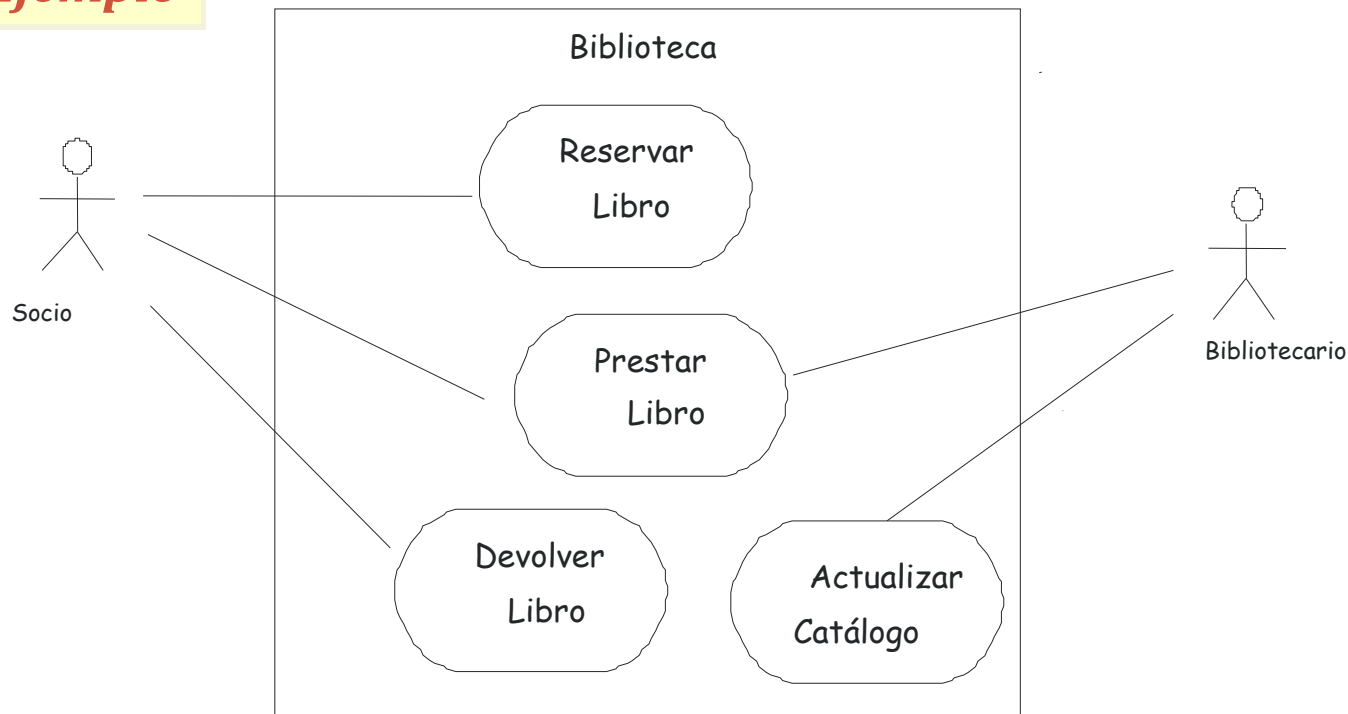


Actores y Casos de Uso

Actor: Entidad (Humana, Dispositivo u Otro Sistemas Software) que intercambia información con el sistema

Caso de Uso Contiene una secuencia de transacciones que intercambian los actores y el sistema cuando se desea ejecutar cierta funcionalidad del mismo

Ejemplo



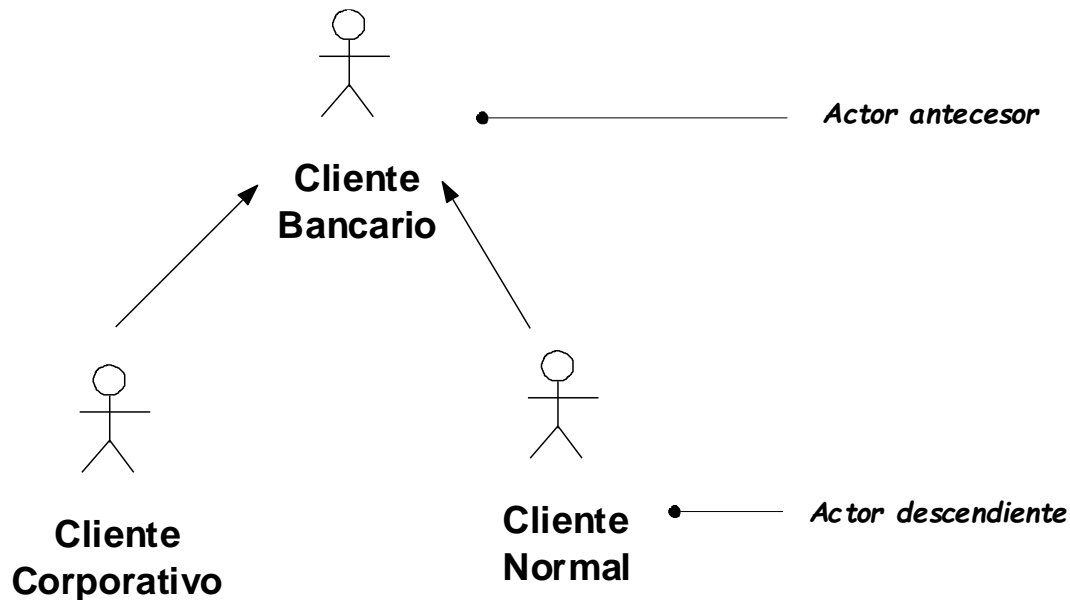
Plantillas de descripción

Los casos de uso se describen utilizando plantillas en lenguaje natural.

| | |
|---------------------------|----------------|
| <i><u>Caso de Uso</u></i> | |
| <i>Actores</i> | |
| <i>Resumen</i> | |
| <i>Precondiciones</i> | |
| <i>Postcondiciones</i> | |
| <i>Incluye</i> | |
| <i>Extiende</i> | |
| <i>Hereda de</i> | |
| <i>Flujo de Eventos</i> | |
| <i>Usuario</i> | <i>Sistema</i> |
| | |

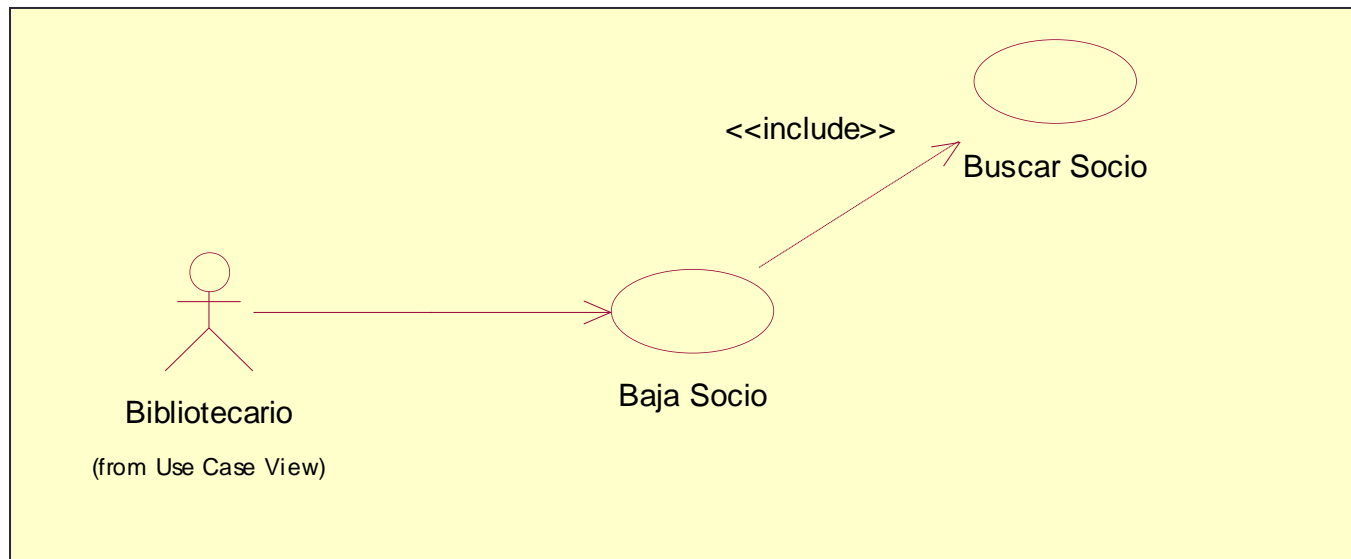
Relaciones entre Actores - Herencia

- La relación de herencia entre actores indica que el actor descendiente puede jugar todos los roles del actor antecesor.



Relaciones entre Casos de Uso - Inclusión

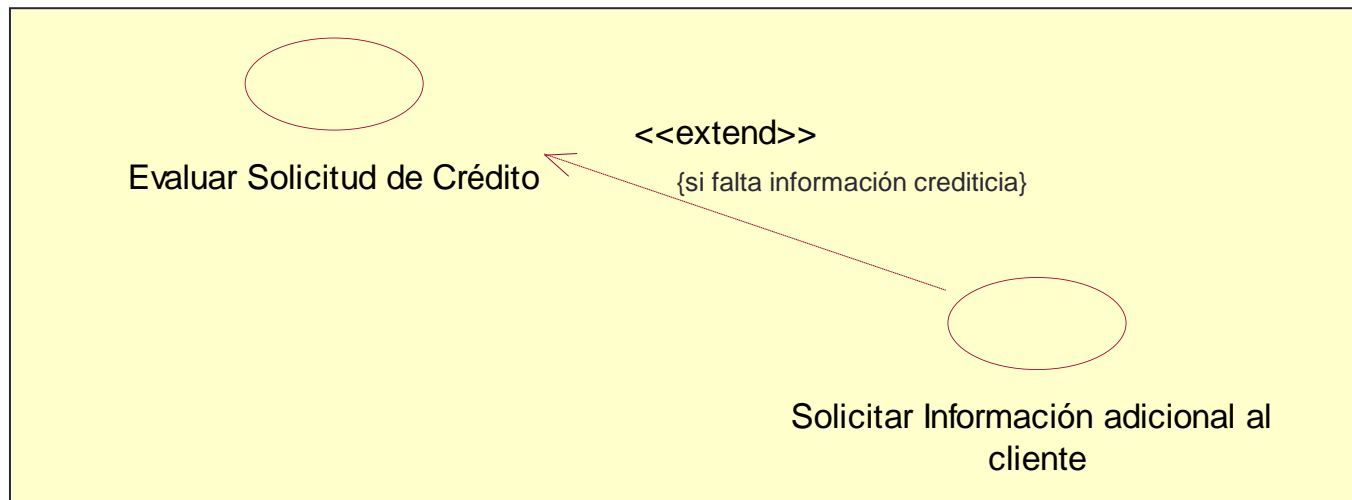
- Un caso de uso A incluye a un caso de uso B, si una instancia de A puede realizar todos los eventos que aparecen descritos en B.



La instanciación de Baja Socio utiliza siempre el flujo de eventos de Buscar Socio

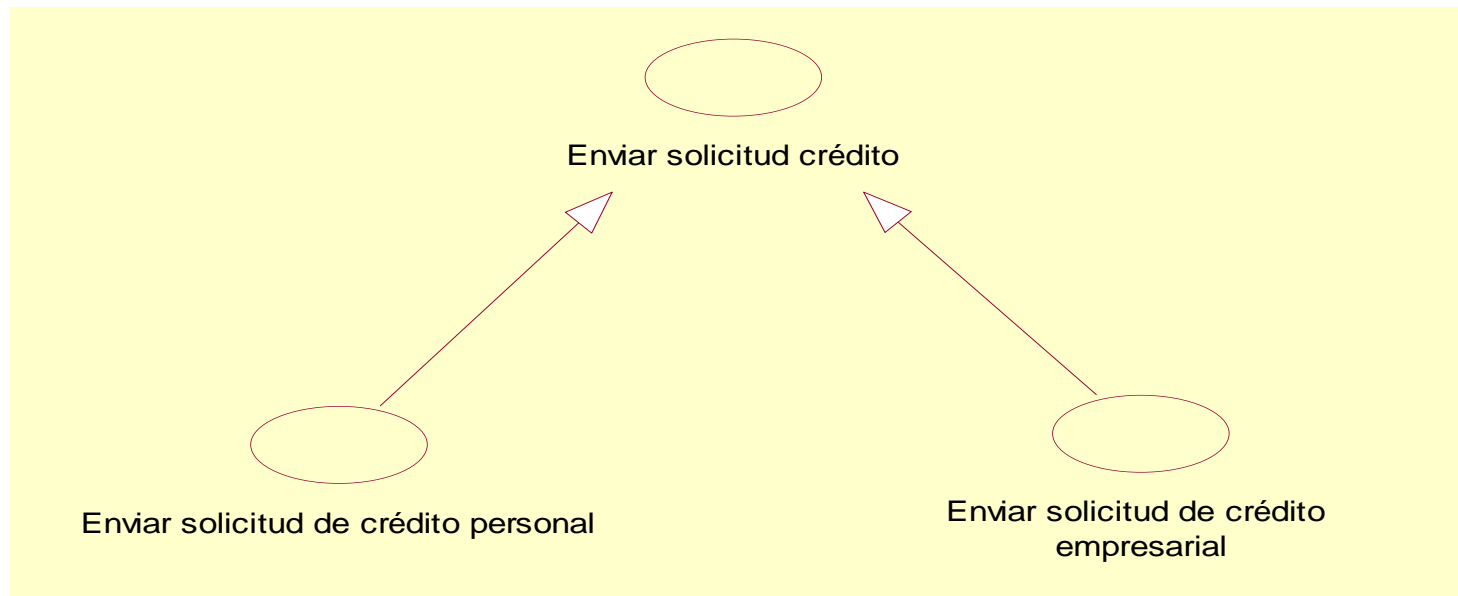
Relaciones entre Casos de Uso - Extensión

- Un caso de uso B extiende a un caso de uso A, si en la descripción de A figura una condición cuyo cumplimiento origina la ejecución de todos los eventos que aparecen descritos en B.



Relaciones entre Casos de Uso - Herencia

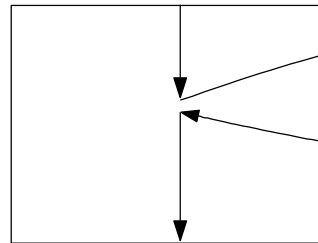
- Un caso de uso B especializa a un caso de uso A, si el flujo de eventos de B es un refinamiento del flujo de eventos asociado a A
 - Similar a la herencia OO (permite separar un patrón de interacción genérico (caso padre) de un patrón de interacción más específico (caso descendiente).



Relaciones entre Casos de Uso

- Inclusión: En la descripción del caso de uso A se incluye una referencia a B

Flujo de eventos del caso de uso A

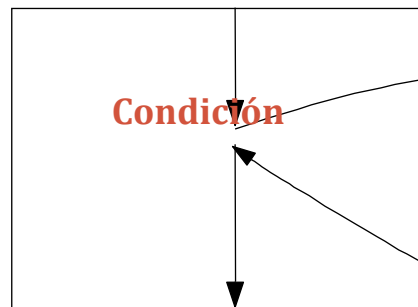


Flujo de eventos del caso de uso B



- Extensión: Equivale a una inclusión más una condición

Flujo de eventos del caso de uso A



Flujo de eventos del caso de uso B

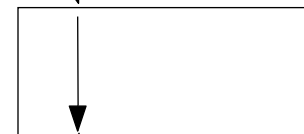


Diagrama de Casos de Uso

- Estructurado en tres capas:
 - Diagrama de contexto y Diagrama inicial.
 - Plantillas de descripción.
 - Diagrama estructurado o Modelo de Casos de Uso.

Diagrama de contexto

- Muestra los límites del sistema y los actores que interactuarán con el mismo.

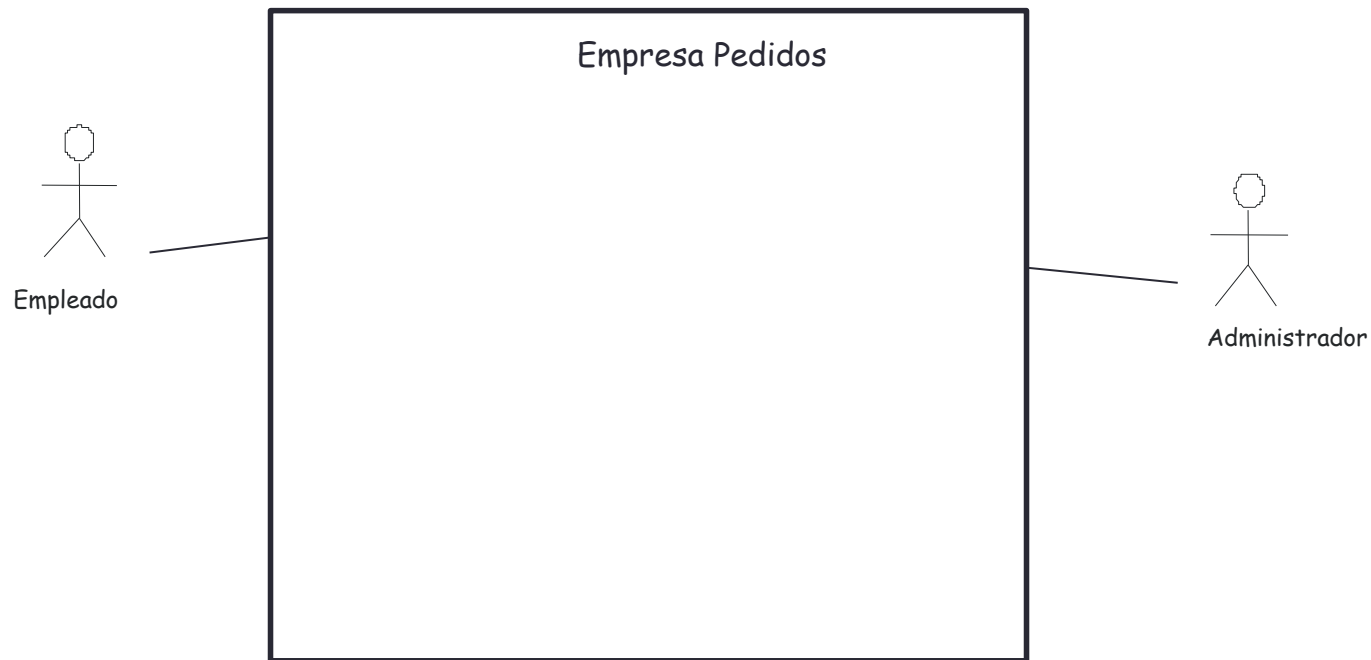
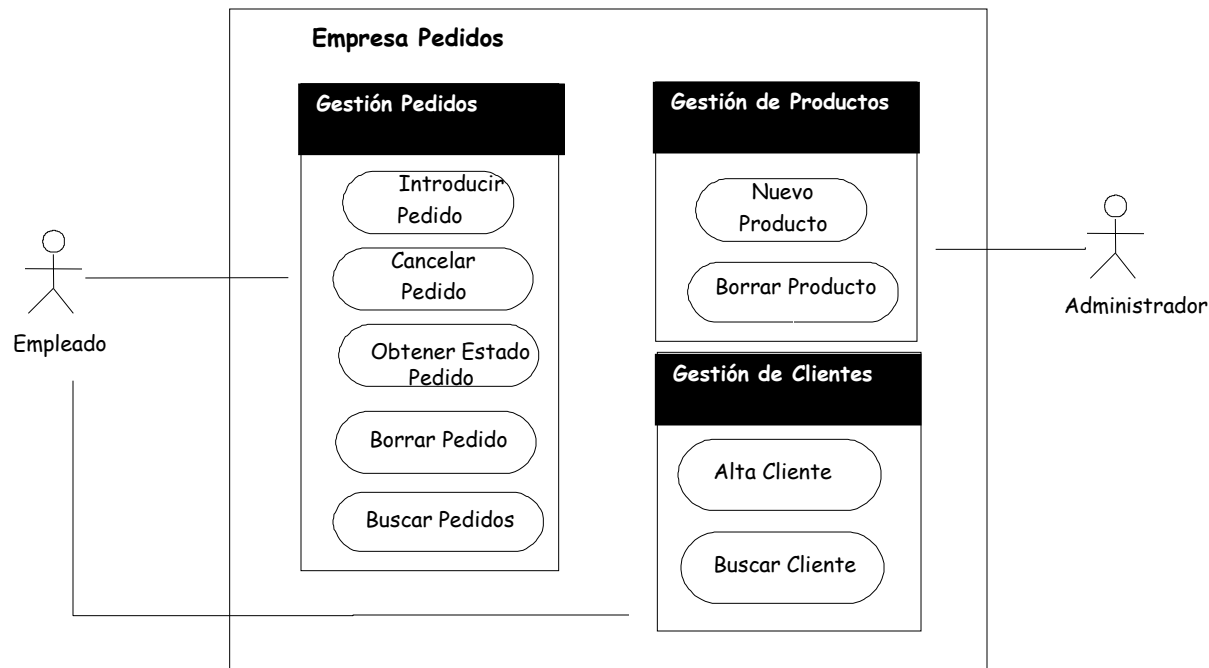
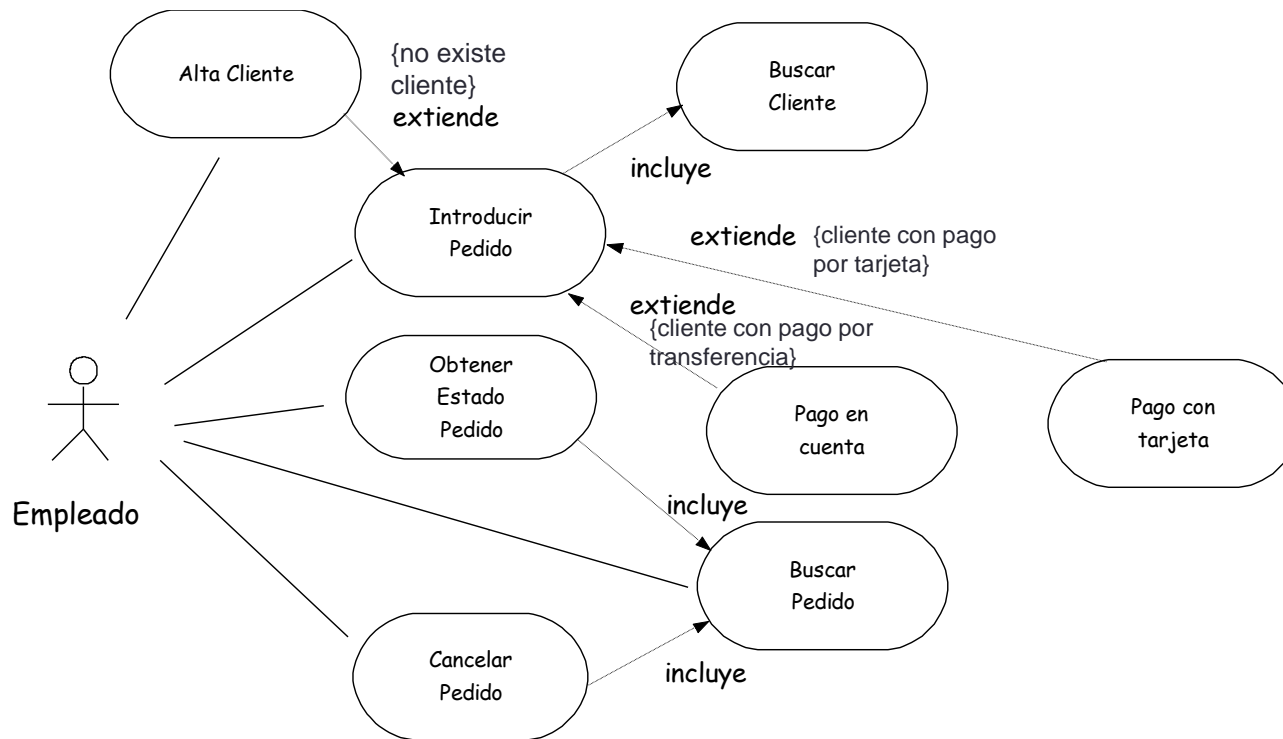


Diagrama Inicial

- Contiene la agrupación jerárquica de los distintos casos de uso:.



Modelo de Casos de Uso

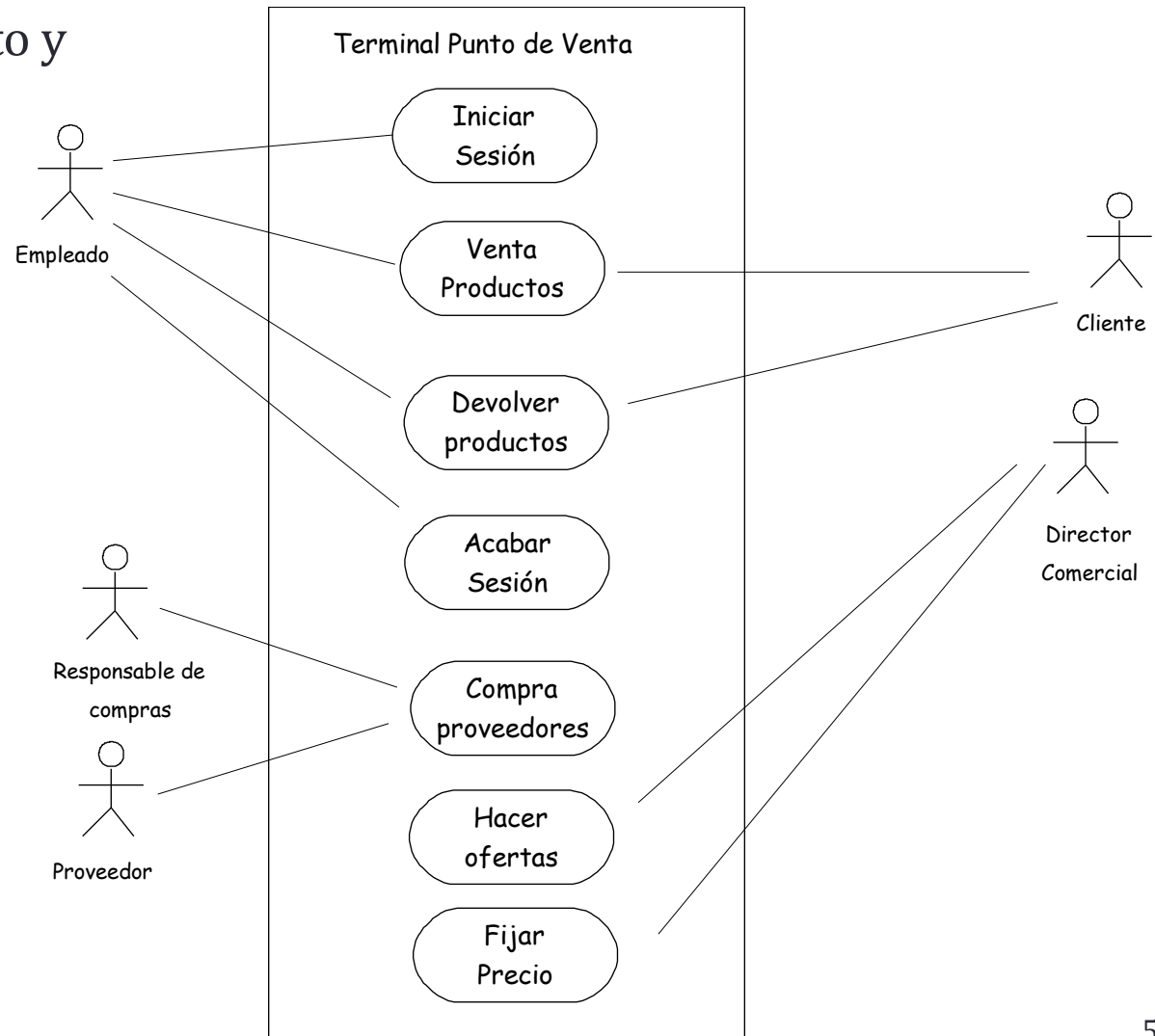


(... el modelo no está completo)

Ejemplo TPV

Terminal Punto de Venta

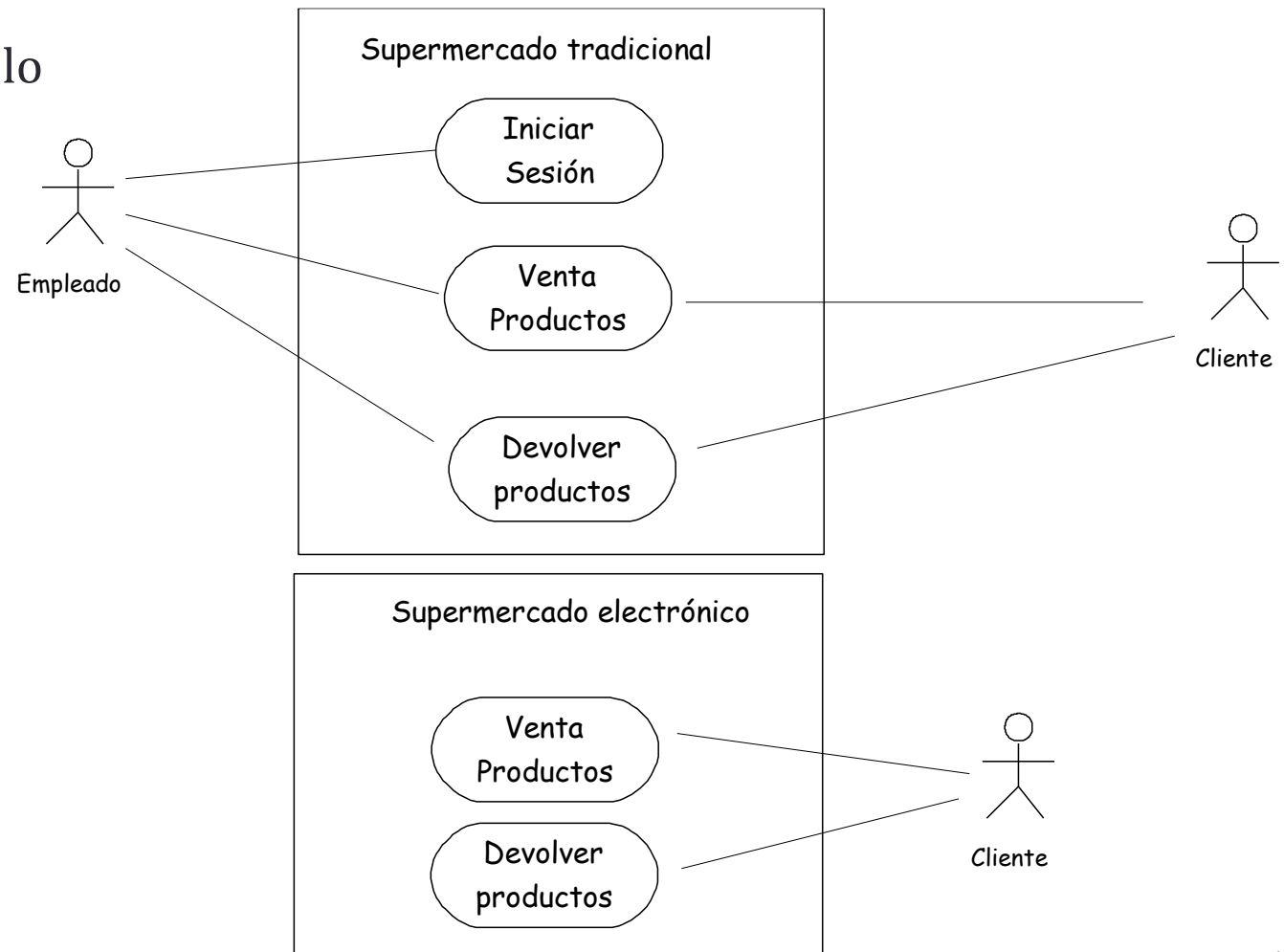
Diagrama de contexto y
Diagrama inicial.



... Ejemplo TPV

Terminal Punto de Venta

Variaciones
sobre el ejemplo

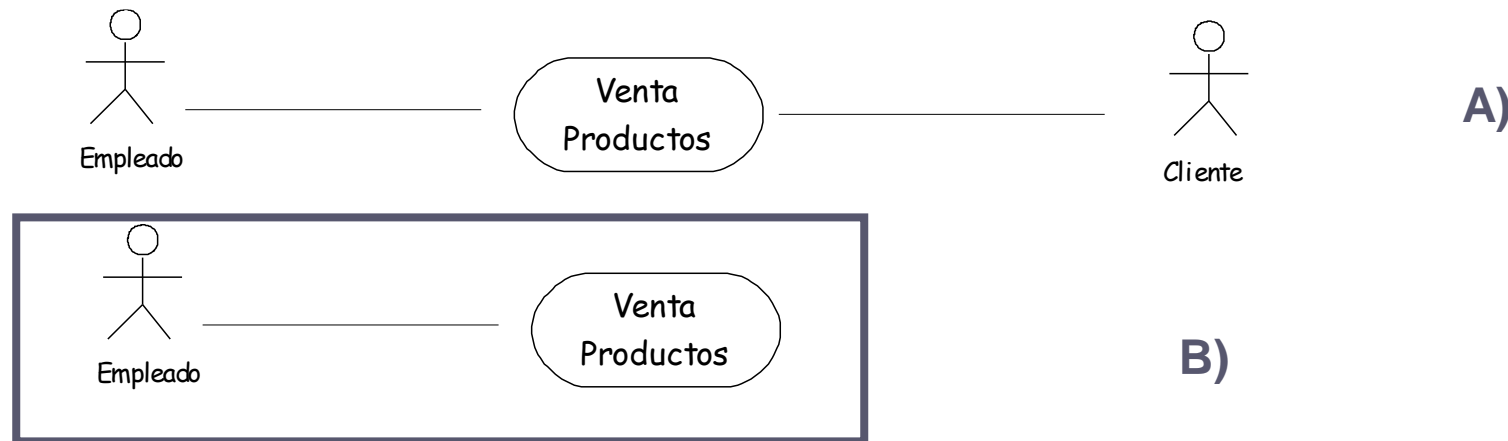


... Ejemplo TPV

Terminal Punto de Venta

Variaciones sobre el ejemplo

Si únicamente se desea mostrar la interacción de los actores con el sistema informático.



... Ejemplo TPV

Terminal Punto de Venta

Plantilla de descripción

**B)**

| | |
|------------------------|---|
| Caso de uso | Venta de Productos |
| Actores | Empleado (iniciador) |
| Propósito | Capturar una venta y su pago en efectivo |
| Resumen | Un cliente llega a la caja con productos para comprar. El empleado registra los productos y gestiona el pago en efectivo. Al acabar el cliente se va con los productos. |
| Precondiciones | El empleado se ha identificado en el sistema. |
| Postcondiciones | La venta se almacena en el sistema. |
| Incluye | - |
| Extiende | - |
| Hereda de | - |

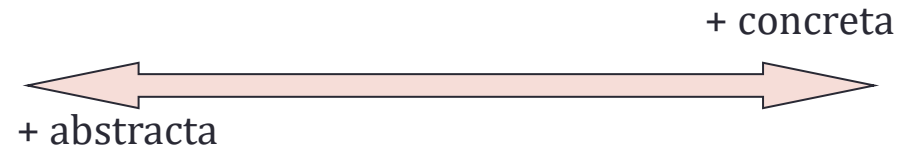
... Ejemplo TPV

Terminal Punto de Venta

... Plantilla de descripción

| Intenciones de usuario | Obligaciones del sistema |
|---|--|
| 1. El empleado indica que comienza una nueva venta. | 2. El sistema registra el inicio de una venta |
| 3. El empleado introduce el código de cada producto y la cantidad | 4. El sistema determina el precio del producto y añade la información a la cuenta. |
| 5. El empleado indica el fin de la venta | 6. El sistema calcula y muestra el total. |
| 7. El empleado indica el dinero que ha recibido | 8. El sistema calcula y muestra el cambio. Imprime un recibo y registra la venta. |
| Extensiones síncronas | |
| #1. Si en 3 se introduce un código de producto inexistente el sistema genera un mensaje de error. | |
| #2. En 7 el empleado puede cancelar la venta. | |
| Extensiones asíncronas | |
| Ninguna | |

Estilos de descripción



**Obtener Efectivo,
(Caso de uso concreto)**

| Acción de Usuario | Respuesta del Sistema |
|---------------------|-----------------------------|
| Insertar Tarjeta | |
| | Leer cinta magnética |
| | Solicitar PIN |
| Introducir PIN | |
| | Verificar PIN |
| | Presentar menú de operación |
| Pulsar tecla | |
| | Presentar menú de cuenta |
| Pulsar tecla | |
| | Preguntar cantidad |
| Introducir cantidad | |
| | Hacer eco de cantidad |
| Pulsar tecla | |
| | Devolver tarjeta |
| Recoger tarjeta | |
| | Dispensar dinero |
| Recoger Dinero | |

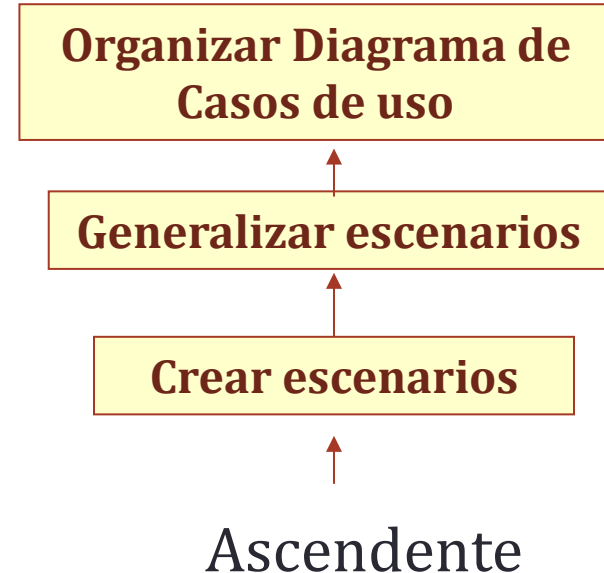
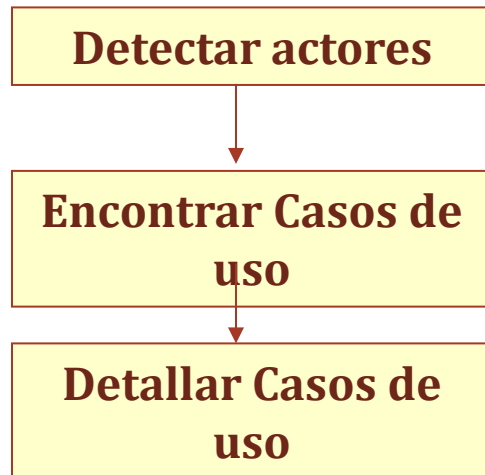
**Obtener Efectivo,
(Caso de uso esencial)**

| Intenciones de Usuario | Responsabilidades Sistema |
|------------------------|----------------------------|
| Identificarse | |
| | Verificar identidad |
| | Ofrecer elecciones de menú |
| Elegir | |
| | Entregar dinero |
| Recoger dinero | |

Construcción del diagrama

- Técnica descendente.
- Técnica ascendente.

Descendente



Construcción del diagrama

Reglas para detectar Actores

- Los usuarios pueden jugar varios roles cuando interactúan con el sistema.
 - Un usuario se puede corresponder con varios actores.
- Cualquier grupo o individuo que caiga en alguna de las siguientes categorías:
- ¿Quién usará el sistema?
 - ¿Quién instalará el mismo?
 - ¿Quién hará labores de mantenimiento?
 - ¿Quién lo apagará?
 - ¿Qué otros sistemas se comunicarán con éste?
 - ¿Quién obtiene información?
 - ¿Quién proporciona información?

Construcción del diagrama

Reglas para identificar Casos de Uso

- Prestando atención a los actores:
 - ¿Cuáles son las tareas que los actores quieren que el sistema realice para ellos?
 - ¿Podrá un actor crear, almacenar, cambiar o borrar datos del sistema?
 - ¿Será necesario que un actor informe al sistema sobre cambios que han ocurrido en el exterior del mismo?
 - ¿Será necesario que el actor sea informado sobre ciertas ocurrencias o cambios dentro del sistema?

Las respuestas a cada una de las preguntas anteriores representan flujos de eventos que identifican casos de uso candidatos.