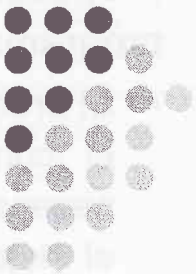
Tema 2

Análisis y Diseño Orientado a Objetos



Análisis textual de Abbot

- Técnica informal que se maneja también en análisis estructurado.
- Su objetivo es obtener una primera lista de clases, relaciones, atributos, métodos, etc. partiendo de una descripción textual del problema.
- Pero ¡ojo!, **sólo sirve de primera aproximación.**

Elemento textual:	Puede corresponder a:
Nombre propio	Instancia o valor de atributo
Nombre común	Clase o rol
Verbo posesivo	Agregación o asociación
Verbo de clasificación	Herencia
Otros verbos	Operación
Adjetivos	Valor, atributo o clase
Frases adjetivas	Asociación o método

Técnica CRC (I)

- Es una técnica informal que permite comprobar el funcionamiento de un sistema.
- Propuesta por Cunningham para facilitar la transición de la programación estructurada a la orientada a objetos:
  - El problema fundamental en OO es conseguir que se abandone el conocimiento global de control (función main()) que es necesario en la programación estructurada y se piense en que el sistema es posible con el conocimiento local de los objetos para llevar a cabo las tareas.
- Utiliza unas tarjetas que se denominan CRC (Class, Responsability, Collaboration).

Técnica CRC (II)

- El contenido mínimo de las tarjetas CRC es:
  - Nombre de la clase.
  - Responsabilidades.
  - Colaboradores.
- Respectivamente, CRC:

Clase	
Responsabilidades	Colaboradores

Sesión CRC

- El número de participantes se aconseja sea de 4-6 y todos deben estar familiarizados con los requisitos del sistema (pueden ser analistas o usuarios).

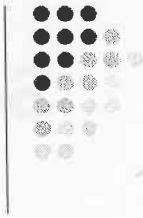
- Las fases son:
  1. BRAINSTORMING.
  2. FILTRADO.
  3. IDENTIFICACIÓN DE ESCENARIOS.

Estrategias de análisis

- Las estrategias ayudan a determinar qué objetos se modelizan en el sistema y sus responsabilidades. Es decir:

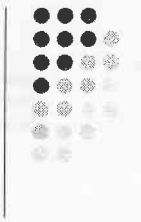
  1. Objetos.
  2. Responsabilidades:
     i. Lo que sé.
     ii. Lo que hago.
     iii. A quién conozco.

Seleccionar objetos (I)



- **SELECCIONAR ACTORES DEL SISTEMA**
  - Persona, organización o entidad externa en general, que participa en el sistema. Puede desempeñar varios papeles (roles).
  - Ejemplos: Persona, Organización, ...
- **SELECCIONAR ROLES DE LOS ACTORES**
  - Cada uno de los papeles que puede jugar un actor.
  - Ejemplos: Cliente, Distribuidor, Suministrador, Vendedor, ...

Seleccionar objetos (II)



- **SELECCIONAR LUGARES-CONTENEDORES**
  - Donde pueden hallarse las cosas u objetos que pueden contener otros objetos.
  - Ejemplos: Almacén, Zona, Armario, Estante, ...
- **SELECCIONAR ELEMENTOS**
  - Tangibles o no: dispositivos electrónicos, otros sistemas, etc.
  - Ejemplos: Pedido, Factura, Stock, ...

Seleccionar objetos (III)

- **SELECCIONAR TRANSACCIONES**
  - Momento o intervalo de tiempo del que hay que recordar algo o sobre el que hay que hacer algo.
  - Ejemplos: Petición, Entrega, Compra, Pago, ...
- **SELECCIONAR DESCRIPTORES DE ÍTEMS**
  - Elementos con valores y acciones aplicables a un conjunto de ítems específicos. Conviene pensar por separado en cuanto a la definición y el uso.
  - Ejemplos: Catálogo-Elemento, Libro-Ejemplar, ...

Seleccionar objetos (IV)

- **SELECCIONAR TIPOS DE OBJETOS**
  - Para encontrar clases adicionales, pensar en si se puede y es útil definir una herencia.
  - Para cada clase se podría hacer:
    1. Ver si puede ser una superclase e intentar nombrar sus posibles subclases.
    2. Ver si puede ser una subclase e intentar identificar sus posibles superclases.
  - Usar la herencia cuando:
    1. Podamos clasificar un objeto dentro de la jerarquía definida y
    2. Siempre permanezca en ella.

Seleccionar objetos (V)

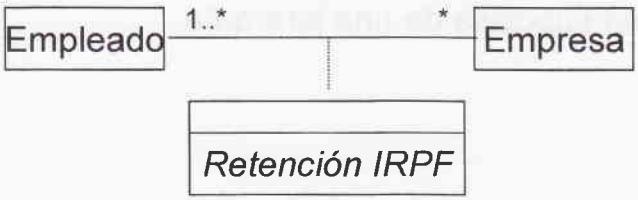
- **BASARSE EN ANALOGÍAS**
  - Buscar sistemas con propósitos similares.
  - Generalizar el propósito del sistema y buscar mini-mundos análogos.
  - Seleccionar los objetos del sistema análogo y seleccionar los que podrían ser objetos de nuestro sistema (por analogía).
  - Añadir esos objetos "metáforas" a nuestro sistema.
- **SELECCIONAR PARA REUTILIZAR**
  - Para cada clase que se modele, buscar el nombre más adecuado; considerando sinónimos, generalidad en los nombres, metáforas de otros sistemas, ...

Lo que sé (I)

- **ATRIBUTOS DE LOS OBJETOS REALES**
  - De los atributos del mundo real, seleccionar los que describen al objeto según las responsabilidades que deba tener en el sistema considerado.
- **ATRIBUTOS SÍ/NO**
  - Atributos que únicamente admiten los valores sí o no: ¿es un atributo o un valor de otro atributo?
  - Ejemplo:
    - Prestable: [sí|no] → Estado: [depósito|prestado|retirado].

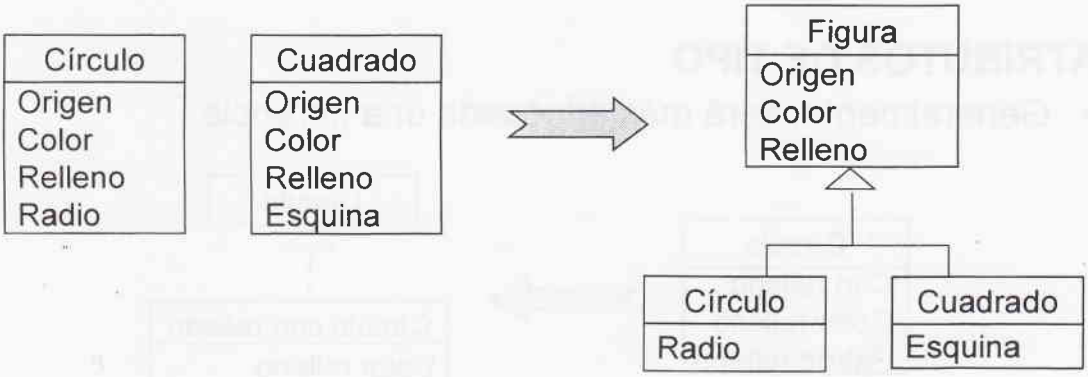
Lo que sé (II)

- **¿ES MÍO?¿ES TUYO?¿ES DE ALGO ENTRE TÚ Y YO?**
  - Cada atributo debe estar sujeto a las siguientes preguntas:
    - ¿Describe a este objeto?: el atributo es de esta clase.
    - ¿Describe a alguien conocido?: añadir el atributo a esa clase.
    - ¿Describe algo en medio que ni de uno ni de otro?: añadir una clase entre las dos y disponer ahí el atributo.
  - Ejemplos: retención IRPF en varios empleos, horas de dedicación en cada proyecto, ...



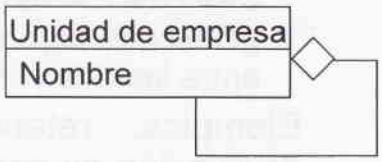
Lo que sé (III)

- **ATRIBUTOS COMUNES**
  - Atributos comunes con el mismo nombre y significado pueden indicar la existencia de una herencia.

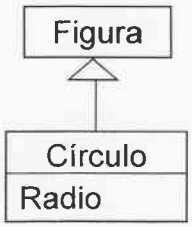


Lo que sé (IV)

- **HACER REFLEXIVAS RELACIONES DE AGREGACIÓN**
  - Cuando en una relación de agregación hay los mismos atributos para el todo y la parte, convertir en una clase con una relación de agregación reflexiva.
  - Ejemplo: Empresa-Área-Departamento ...



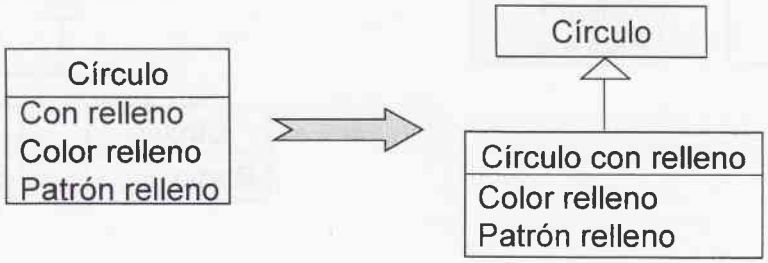
- **ATRIBUTOS APLICABLES PARCIALMENTE**
  - Llevarlos a una subclase de una jerarquía.



Lo que sé (V)

- **OBJETOS SIN ATRIBUTOS**
  - Este tipo de objetos estarán bien en general si tienen relaciones con otros objetos. Si se trata de una encapsulación de funcionalidad que pueden hacer otros objetos es mejor repartirla entre esos.

- **ATRIBUTOS DE TIPO**
  - Generalmente, será más adecuado una herencia.



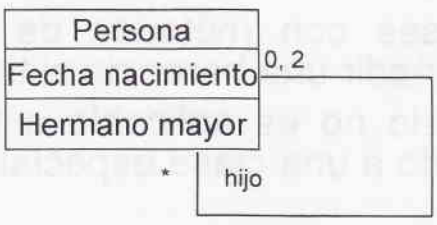
Lo que hago (I)

- **LO QUE DEBO HACER**
  - El objeto debe hacer las cosas que hace el objeto del mundo real del que es abstracción. Se debe encapsular atributos y operaciones relacionadas.
  - Los objetos del sistema software también deben hacer las cosas que el sistema es responsable de hacer con respecto a ellos.
  - Las operaciones deben de localizarse en el objeto que tiene los atributos sobre los que actúan.

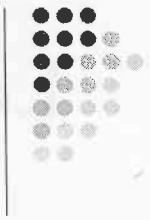
- **SERVICIOS BÁSICOS (de atributos y conexiones)**
  - Los servicios básicos relacionados con los atributos y las relaciones no se ponen en general, salvo avanzado el diseño.
  - Ejemplos: add, remove, get, set, …

Lo que hago (II)

- **¿QUÉ PUEDO HACER?**
  - Para cada objeto se debe pensar en lo que puede hacer, basándose en lo que sabe y a quien conoce, antes de añadir nuevos elementos al modelo.
  - En el ejemplo siguiente no se necesita identificar una relación extra de hermano mayor para saber el mayor de una serie de hermanos.



Lo que hago (III)



- **UBICAR SERVICIOS**
  - Las cosas deben hacerse por uno mismo o si no se puede, pedir colaboración con mensajes muy simples entre objetos.
  - Si tenemos una agregación:
    - El todo opera sobre sus propios atributos.
    - El todo pide a las partes que hagan algo útil por ella (cuanto más mejor).
  - Ejemplo:
    - Factura, Líneas de Factura e Imprimir().
    - La factura imprime sus datos, pero cada línea sabe imprimirse a sí misma y lo hace a petición de la factura.

Lo que hago (IV)

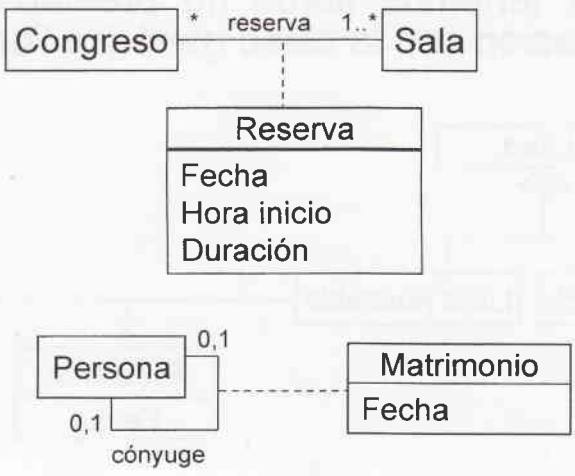
- **EXCESO DE TRABAJO**
  - Los mensajes deben ser simples: cuidado con los objetos que interaccionan con muchos objetos del dominio. Debe distribuirse la responsabilidad.
  - En lugar de pedir un dato, pedir algo útil.
- **SERVICIOS COMUNES Y PARCIALMENTE APLICABLES**
  - Si hay clases con métodos de nombre y significados comunes, añadir una herencia si tiene sentido y es útil.
  - Si un método no es aplicable a todos los objetos de la clase, llevarlo a una clase especializada.

A quién conozco (I)

- **EN GENERAL**
  - Modelizar asociaciones para:
    - Conocer directamente al destinatario de mensajes o
    - Responder preguntas sobre los objetos que están relacionados.
- **RELACIONES A-B-C**
  - Modelizar la asociación A-C si A y C pueden existir sin B y se necesita conocer la asociación incluso si B no existe.
- **RELACIONES 1:1**
  - A menos que no sea posible, combinar en una clase.

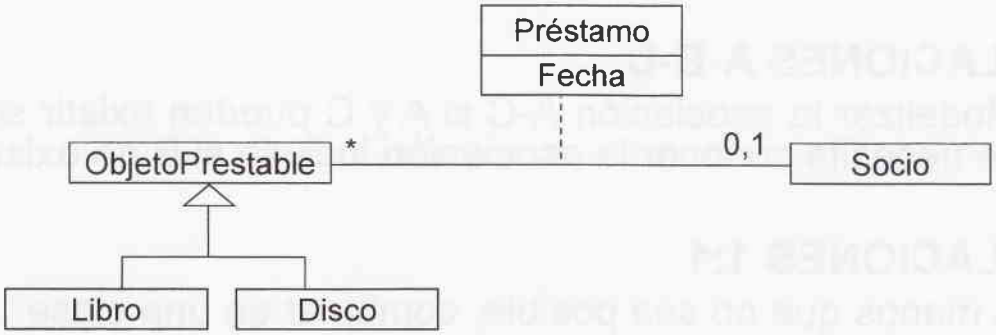
A quién conozco (II)

- **RELACIONES N:N Y REFLEXIVAS**
  - Pensar si puede haber un objeto de tipo transacción entre las clases asociadas.



A quién conozco (III)

- **RELACIONES COMUNES**
  - Añadir una herencia si tiene sentido y se puede.
  - Ejemplo: en vez de tener dos relaciones para préstamo de libros y de discos podemos tener una de la siguiente manera:



A quién conozco (IV)

- **RELACIONES NO COMUNES**
  - Para las relaciones que sólo sean aplicables a ciertos objetos de la clase la asociación debe implicar a una subclase de una jerarquía de herencia.
  - Ejemplo: si tenemos libros no prestables, no se debe poner la relación con la clase genérica libro: