

## Modelo alternativo de análisis: Modelo de Jacobson

- Modelo de análisis de Jacobson o análisis de la robustez (“Robustness Analysis”)
- Es un nivel de diseño intermedio entre la etapa de Captura de requerimientos y la de diseño
- Ivar Jacobson (uno de los creadores de UML)
- Semejanzas con el patrón “modelo-vista-controlador”

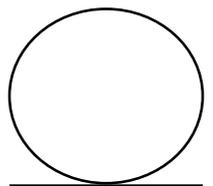
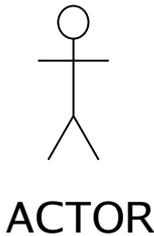
## Análisis de Jacobson

- Proporciona un diseño preliminar
- Puede ayudar a descubrir la necesidad de clases adicionales
- Proporciona una prueba de completitud a los casos de uso, antes de pasar al diseño
- Proporciona un diseño preliminar de la arquitectura del SI

## Análisis de Jacobson

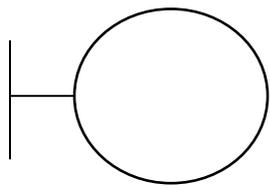
- No forma parte de UML
- No siempre es usado (no adecuado a OO)
- Doug Rosenberg
  - “Use Case Driven Object Modelling with UML”
  - Recomienda hacer el análisis de la robustez antes del diagrama de interacción (diseño)
- Crea responsabilidades (métodos) antes del diseño

# Diagrama de Jacobson



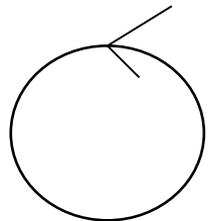
Entidad

Representa  
datos  
almacenados



Interfaz

Representa  
una interfaz  
del sistema



Control

Representa  
transferencia  
de información

frontera

Patrón  
de  
diseño



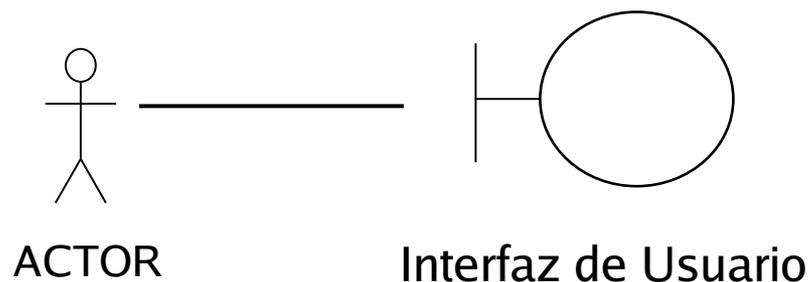
## Componentes de un modelo de Jacobson

- **Entidades**
  - Modelan información perdurable p.e. entre casos de uso
  - “modelo” que captura los datos ...
- **Interfaz**
  - Transporta la acción del actor a los eventos del sistema
  - Transporta al actor los eventos del sistema
  - Cada actor puede tener su conjunto de interfaces
  - ... que pueden ser “vistos” de múltiples formas ...
- **Control**
  - Unen otros componentes para formar un caso de uso
  - ... mediante los “controladores”, que proporcionan formas de actualizar y extraer información del modelo.

## Clases Interfaz o Frontera

- Modelan la interacción entre el sistema y los actores
- Clarifican los requisitos en la frontera entre sistema y usuarios. Cambios en los interfaces de usuario, de comunicación, etc. afectan a las clases frontera
- Representan abstracciones de ventanas, formularios, sensores, terminales y APIs (Application Program Interfaces)
- Deben estar asociados a un actor

## Clases Interfaz o Frontera

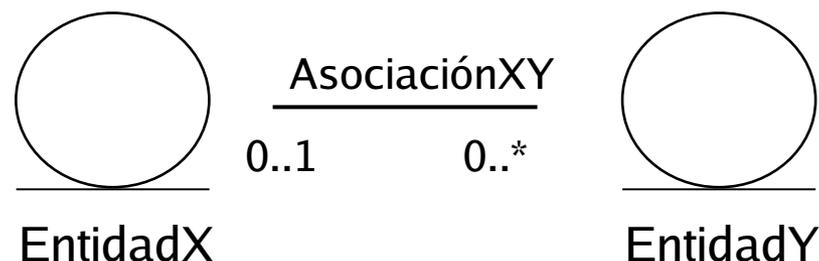


- Como entrada de información permite que el actor
  - Proporcione datos (cajas de texto, menús desplegables, ...)
  - Solicite servicios (pulsando botones, ...)
- Como salida de información
  - Presenta datos al actor (listados, texto, ...)
- Puede conectar con un actor o una clase de control

## Clases Entidad

- Modelan la información y el comportamiento asociado de conceptos (individuos, objetos, eventos) del mundo real
- En la mayoría de los casos las clases entidad se derivan de clases del modelo de dominio
- Las clases entidad muestran la estructura lógica de los datos

## Clases Entidad

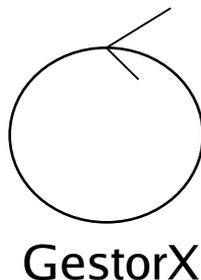


- Las clases entidad (y las asociaciones entre ellas) permiten mostrar la estructura lógica de los datos
- ... pueden servir para modelar la información del SI
- ... según Rosenberg (reglas de robustez) sólo deben conectar con clases de control

## Clases de Control

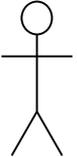
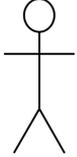
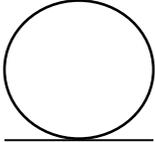
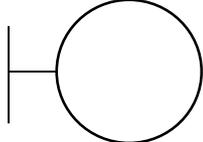
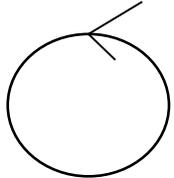
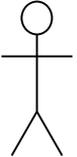
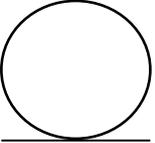
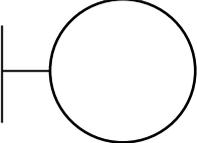
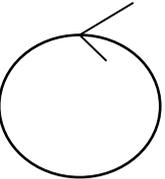
- Modelan la coordinación, secuencia, transacciones y control del flujo de la información
- Representan la lógica del negocio no presente en las clases entidad
- No interaccionan con los actores
- No representan la información persistente del sistema

## Clases de Control

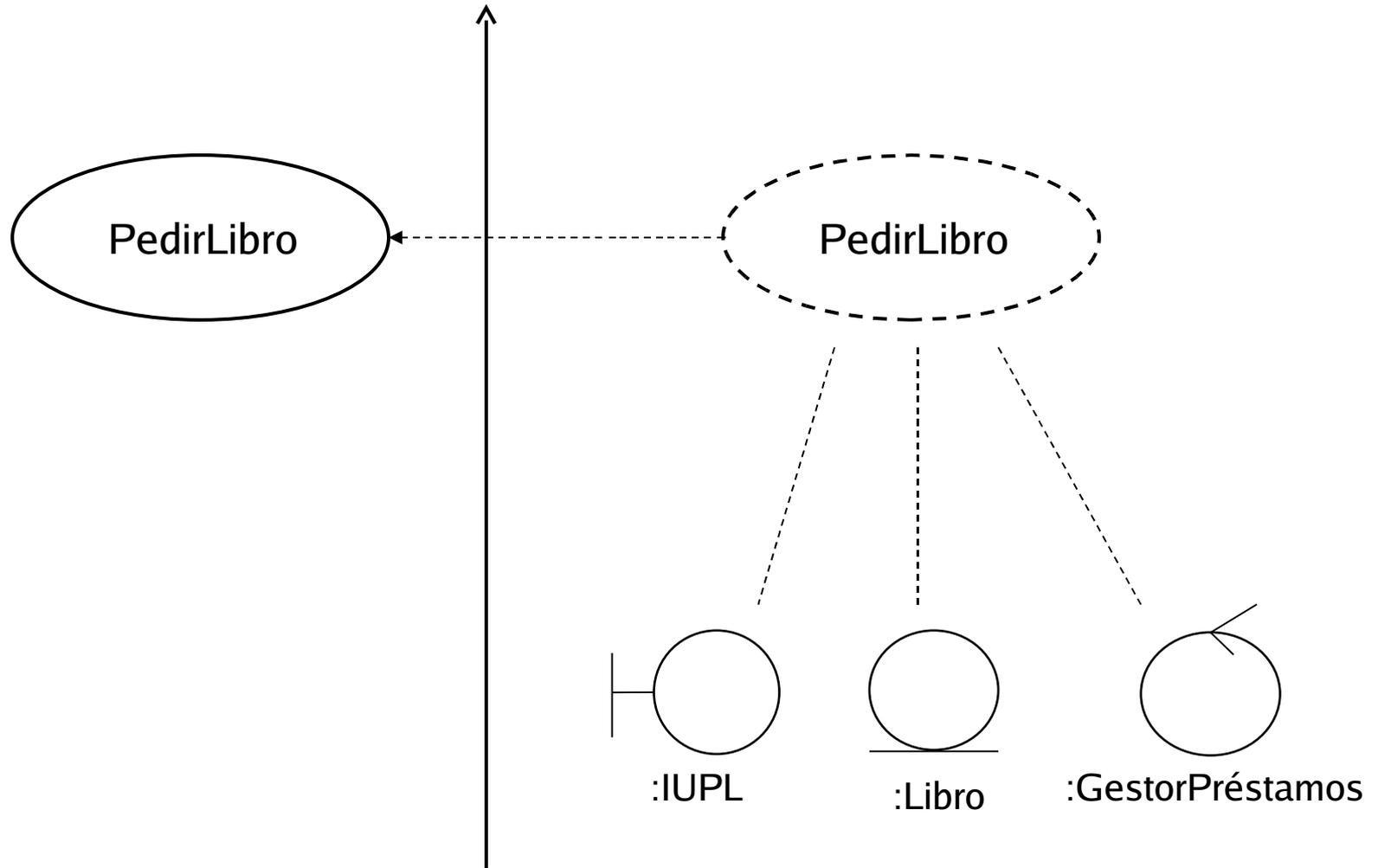


- Una clase de control permite:
  - Buscar una información concreta de una clase conociendo alguno de los valores de sus atributos
  - Crear/modificar/eliminar información
  - Realizar procesos/cálculos relacionados con la lógica del negocio
- No pueden conectarse directamente con los actores

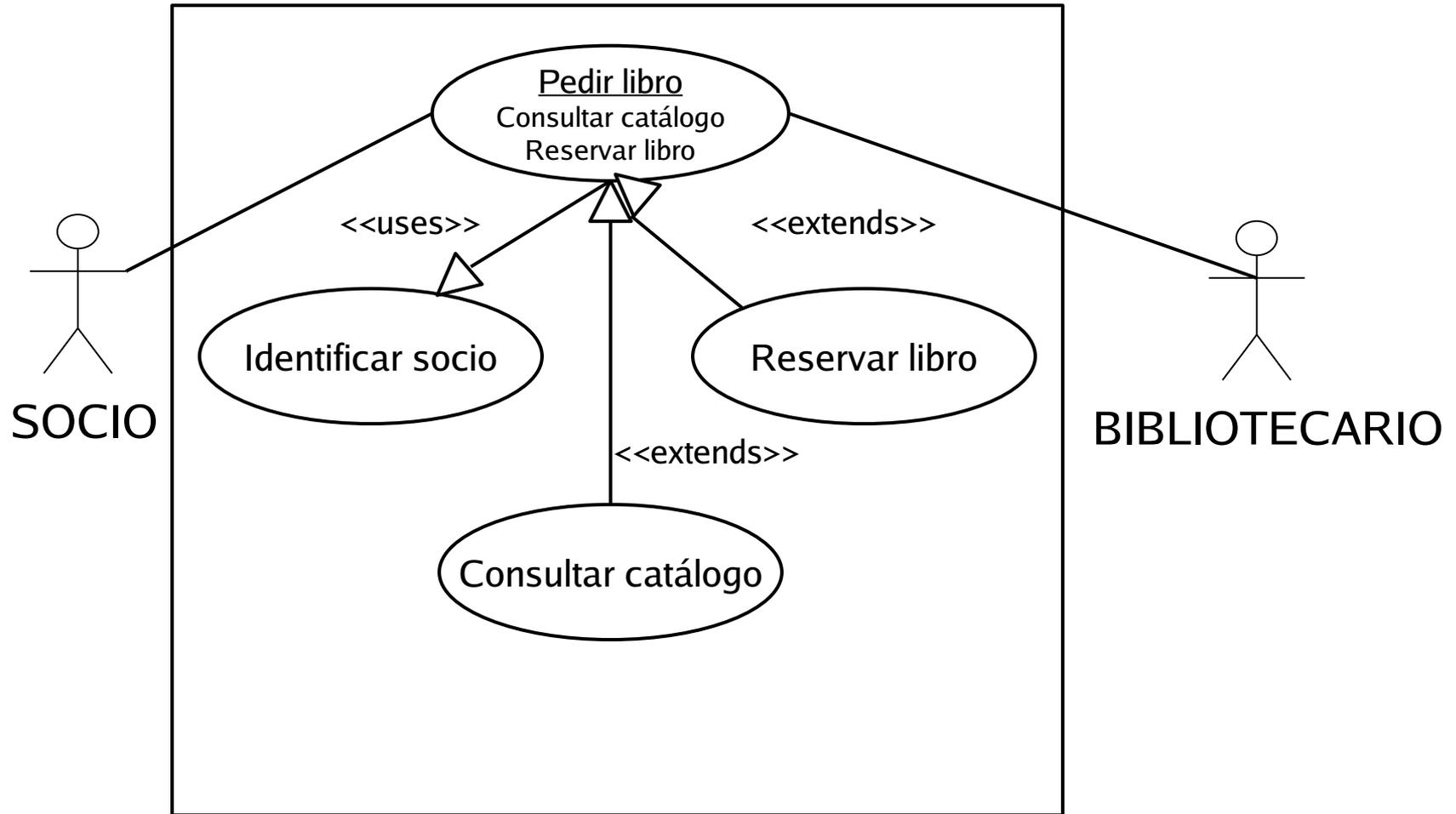
## Reglas de robustez de Rosenberg (puede conectar)

				
ACTOR	ACTOR			
	NO	NO	SI	NO
	NO	NO	NO	SI
	SI	NO	NO	SI
	NO	SI	SI	SI

## Modelo de CU vs. Modelo de Análisis



## Ejemplo: Pedir Libro



## Ejemplo: Caso de uso completo (1)

Caso de uso:     **Pedir libro**

Actores:         Socio, Bibliotecario

Resumen:         Un socio solicita un libro en préstamo al bibliotecario. El bibliotecario verifica y registra el préstamo. Al terminar el bibliotecario le entrega una copia al socio.

Precondiciones: El bibliotecario está identificado.

Postcondiciones: Se registra el préstamo de libro, actualizando los libros prestados del Socio y las copias del libro prestado.

Referencias cruzadas: R1, R2, R3, R4, R5, R7, R8, R9

## Ejemplo: Caso de uso Completo (2): Clases Frontera

### Escenario principal (o curso normal de los eventos)

1. **Socio:** El Socio se identifica y solicita un libro en préstamo al Bibliotecario.
2. **Bibliotecario:** *Identifica al socio.*
3. **Sistema:** *Presenta la información del socio, si es o no profesor y sus libros en préstamo con su fecha de devolución.*
4. **Bibliotecario:** Comprueba que el Socio no tiene libros pendientes de devolución, ni el máximo de libros en préstamo. *Consulta el catálogo.*
5. **Sistema:** *Presenta los libros que cumplen los criterios de búsqueda. La información incluye las copias disponibles, las reservas y el periodo de préstamo y la fecha de devolución de cada copia.*
6. **Bibliotecario:** Verifica las copias disponibles.
7. **Socio:** Confirma el libro buscado y acepta la fecha de devolución.
8. **Bibliotecario:** *Confirma el préstamo.*
9. **Sistema:** Registra el nuevo préstamo con la fecha actual.
10. **Bibliotecario:** Indica al Socio la fecha de devolución del libro.
11. **Socio:** Se marcha con el libro en préstamo.

## Ejemplo: Clases Frontera CU Pedir Libro



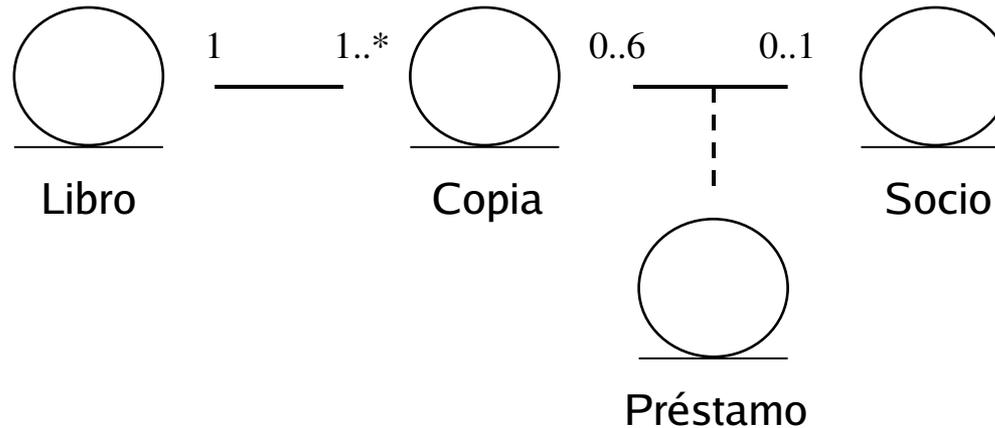
- Permite que el actor bibliotecario
  - Proporcione su identificador de socio
  - Solicitar Consultar Catálogo
  - Solicitar Reservar Libro
  - Confirmar el préstamo
- Presenta información del socio y sus libros en préstamo
- Presenta los libros que cumplen los criterios de búsqueda

## Ejemplo: Caso de uso Completo (2): Clases Entidad

### Escenario principal (o curso normal de los eventos)

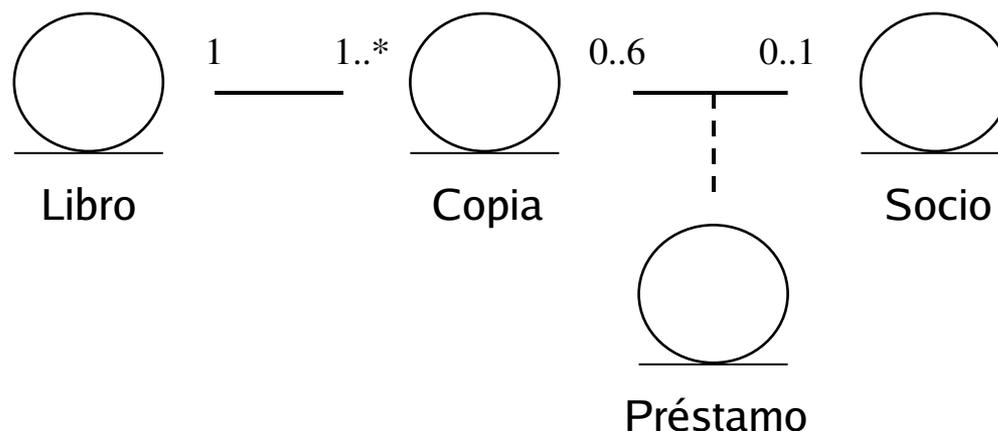
1. **Socio**: El Socio se identifica y solicita un libro en préstamo al Bibliotecario.
2. **Bibliotecario**: Identifica al Socio.
3. **Sistema**: Presenta la información del Socio, si es o no profesor y sus libros en préstamo con su fecha de devolución.
4. **Bibliotecario**: Comprueba que el Socio no tiene libros pendientes de devolución, ni el máximo de libros en préstamo. Consulta el catálogo.
5. **Sistema**: Presenta los libros que cumplen los criterios de búsqueda. La información incluye las copias disponibles, las reservas y el periodo de préstamo y la fecha de devolución de cada copia.
6. **Bibliotecario**: Verifica las copias disponibles.
7. **Socio**: Confirma el libro buscado y acepta la fecha de devolución.
8. **Bibliotecario**: Confirma el préstamo.
9. **Sistema**: Registra el nuevo préstamo con la fecha actual.
10. **Bibliotecario**: Indica al Socio la fecha de devolución del libro.
11. **Socio**: Se marcha con el libro en préstamo.

## Ejemplo: Clases Entidad CU Pedir Libro (1)



- **Atributos**
  - Libro: signatura, ...
  - Socio: dni, /LibrosPrestados, ...
  - Préstamo: fecha, ...
  - Copia: idCopia, ...

## Ejemplo: Clases Entidad CU Pedir Libro (2)



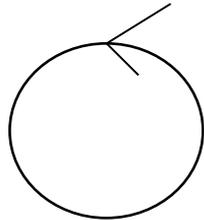
- Una Copia no prestada será un objeto de la clase Copia asociado a un Libro de la misma signatura y que no está asociado a un Socio
- Un Socio tiene el máximo de libros en préstamo si el objeto Socio identificado por dni está asociado al máximo de objetos Copia
- Un nuevo préstamo se añade como objeto de la clase asociación con la fecha actual y asociado a los objetos de Copia no prestada y al Socio

## Ejemplo: Caso de uso Completo (2): Clases Control

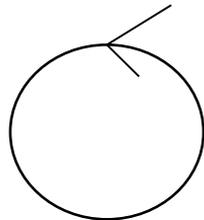
### Escenario principal (o curso normal de los eventos)

1. **Socio:** El Socio se identifica y solicita un libro en préstamo al Bibliotecario.
2. **Bibliotecario:** *Identifica al socio.*
3. **Sistema:** *Presenta la información del socio*, si es o no profesor y sus libros en préstamo con su fecha de devolución.
4. **Bibliotecario:** Comprueba que el Socio *no tiene libros pendientes de devolución, ni el máximo de libros en préstamo. Consulta el catálogo.*
5. **Sistema:** *Presenta los libros* que cumplen los criterios de búsqueda. La información incluye las copias disponibles, las reservas y el periodo de préstamo y la fecha de devolución de cada copia.
6. **Bibliotecario:** Verifica las copias disponibles.
7. **Socio:** Confirma el libro buscado y acepta la fecha de devolución.
8. **Bibliotecario:** Confirma el préstamo.
9. **Sistema:** *Registra el nuevo préstamo* con la fecha actual.
10. **Bibliotecario:** Indica al Socio la fecha de devolución del libro.
11. **Socio:** Se marcha con el libro en préstamo.

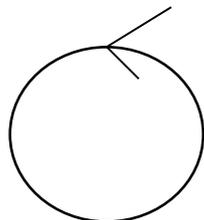
## Ejemplo: Clases Control CU Pedir Libro



GestorLibros



GestorSocios



GestorPrestamos

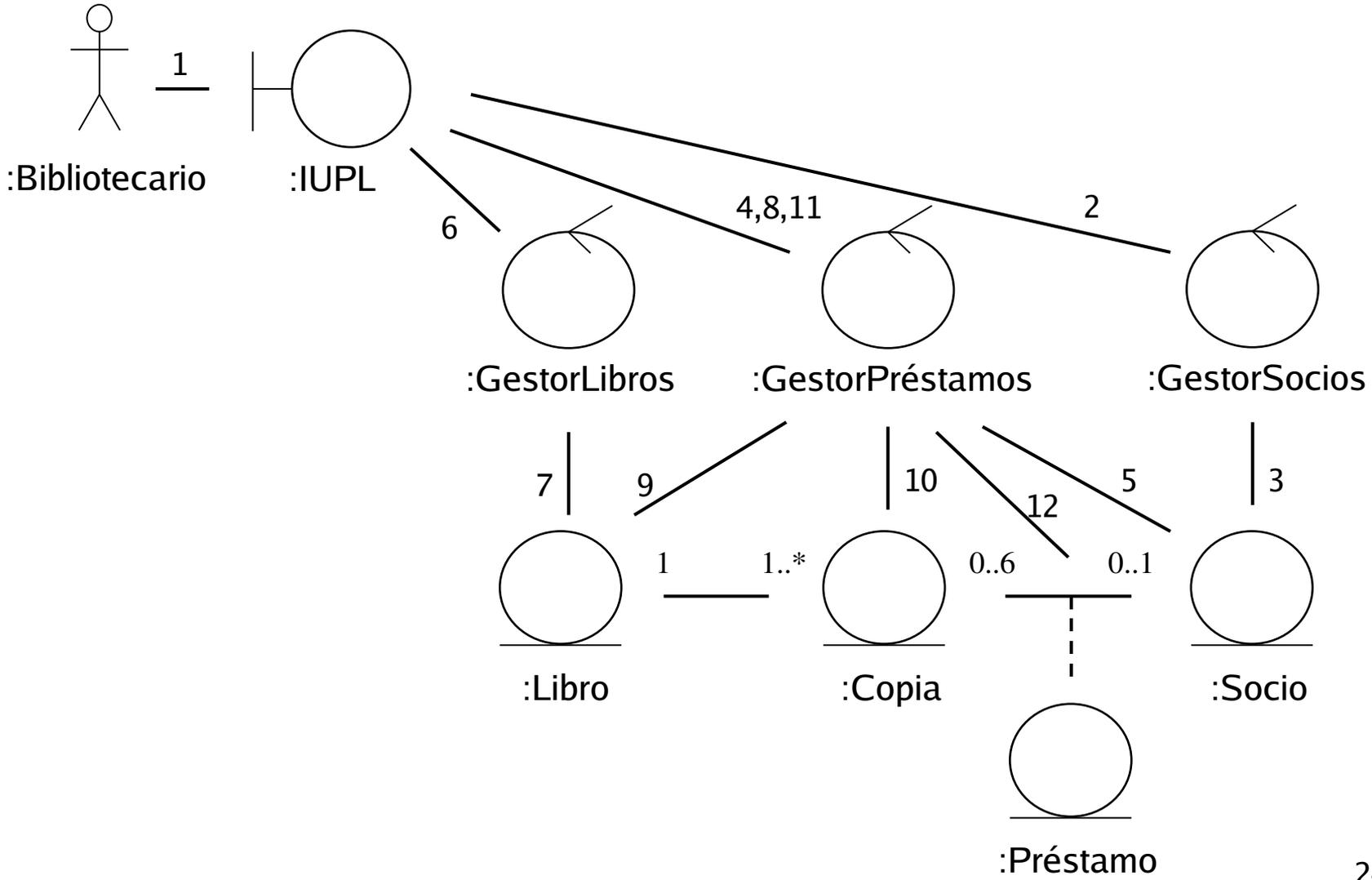
### Responsabilidades

- Buscar Libro por signatura
- Buscar Socio por dni
- Saber si dado un socio ya ha llegado al límite de copias prestadas
- Añadir un nuevo préstamo dado un socio y un libro

## Realización del análisis de un caso de uso

- Indica cómo se realiza/ejecuta el caso de uso
  
- Para cada caso de uso deberemos indicar
  - El diagrama de Jacobson entre las clases
  
  - El curso de los eventos (en el análisis) que explique en Lenguaje Natural el diagrama de Jacobson

# Ejemplo: Diagrama de Jacobson Pedir Libro (1)



## Ejemplo: Diagrama de Jacobson Pedir Libro (2)

- 1: Introducir dni Socio, Introducir Libro por atributo o Confirmar Préstamo
- 2: Buscar Socio por dni
- 3: Obtener Socio
- 4: Comprobar si el Socio ha llegado al límite de préstamos
- 5: Obtener préstamos del Socio por dni
- 6: Buscar Libro por atributo (signatura, título, autor, etc.)
- 7: Obtener Libros
- 8: Obtener Copia no prestada
- 9: Obtener Copias del Libro por signatura
- 10: Verificar si la Copia está prestada
- 11: Registrar préstamo de Copia por Socio
- 12: Crear nuevo préstamo