Análisis

- Análisis en el PUD
- Diagramas de secuencia del sistema
- Contratos
- Object Constraint Language (OCL)
- Modelo alternativo de análisis: Modelo de Jacobson

Iteración en PUD

- Planificación de la Iteración
  - Captura de requisitos:
    - Modelo de casos de uso, Modelo de Dominio, ...
  - Análisis:
    - Diagrama de secuencia del sistema, Contratos, Modelo Conceptual...
  - Diseño:
    - Diagramas de interacción, Diagrama de Clases
  - Implementación:
    - codificación (Clases y métodos)
  - Pruebas:
    - verificación de la implementación
- Evaluación de la iteración

Fases y entregas del Proceso Unificado de Desarrollo

- captura de requisitos: qué SI debemos construir?
  - Modelo de casos de uso, Modelo de Dominio, ...
- análisis: qué debe hacer el SI?
  - Diagramas de secuencia del sistema, Contratos, ...

- diseño: cómo lo debe hacer el SI?
  - Diagramas de interacción, Diagrama de Clases
- codificación:
  - Código Fuente (clases y métodos)
- pruebas:
  - Especificación de las pruebas de funcionamiento
- mantenimiento:
  - Documentación y revisión de todo lo anterior

Dependiente de la tecnología

Construcción incremental e iterativa del SI

- Modelo dinámico del sistema (comportamiento):
  - Captura de requisitos: Modelo de Casos de Uso
  - Análisis: Diagramas de secuencia del sistema, contratos
  - Diseño: Diagramas de interacción
- Modelo estático del sistema (propiedades):
  - Captura de requisitos: Modelo de Dominio
  - Análisis: Modelo Conceptual
  - Diseño: Diagrama de clases

- Implementación: codificación (clases y métodos)

Modelo del comportamiento del sistema

- Diagramas de secuencia del sistema
  - Muestra los eventos entre los actores y el sistema
  - Permiten identificar las operaciones del sistema
- Contratos
  - Describen los efectos de las operaciones del sistema

Diagrama de secuencia del sistema

- Representación que muestra, para un determinado caso de uso, los eventos generados por los actores externos, su orden y los eventos del sistema
- Al sistema se le considera una caja negra
- Los diagramas se centran en los eventos que transcienden las fronteras del sistema y que fluyen de los actores al sistema
- Inicialmente, los diagramas deberían prepararse para el escenario principal de un caso de uso

Diagrama de secuencia del sistema

- Objetivo
  - Identificar los eventos y las operaciones (comportamiento) del sistema
- Partimos de los casos de uso
  - Describen cómo interaccionan los actores con el sistema
  - Los actores generan eventos hacia el sistema que requieren de la ejecución de alguna operación como respuesta
- Definimos un diagrama de interacción para cada curso relevante de los eventos de un caso de uso mostrando:
  - Los eventos generados por los actores externos y su orden
  - Los eventos internos del sistema (operaciones) que resultan de la invocación

Diagrama de secuencia del sistema

- Crear un diagrama de secuencia del sistema para cada caso de uso.
- Cada evento en el diagrama debe corresponder a una interacción con el sistema especificado en el caso de uso completo

- Dibujar una línea vertical que representa el **sistema**
- Dibujar una línea para cada **actor** que interacciona **directamente** con el sistema
- A partir del curso de eventos de los casos de uso, **identificar** y mostrar los **eventos** externos generados por los actores
- Para identificar los eventos del sistema es necesario delimitar claramente la **frontera** del sistema

Ejemplo TPV: caso de uso completo (1)

Caso de uso: **Comprar productos**

Actores: Cliente, Cajero (principal)

Resumen: Un Cliente llega a la caja registradora con los artículos que desea comprar. El Cajero registra los artículos y recibe un pago. Al terminar la operación, el Cliente se marcha con los productos comprados.

Precondiciones: El Cajero está identificado.

Postcondiciones: Se registra la venta completa, su importe y los impuestos. Se actualiza el inventario.

Referencias: R1.1, R1.2, R1.3, R1.4, R1.5, R1.7

Ejemplo TPV: caso de uso completo (2)

Escenario principal (o curso normal de los eventos):

1. **Cliente**: Llega a un TPV con productos que desea comprar.
2. **Cajero**: Comienza una nueva venta.
3. **Cajero**: Introduce el identificador del artículo. Si hay varios productos de una misma categoría, el Cajero también puede introducir la cantidad.
4. **Sistema**: Registra la línea de la venta, y presenta la descripción del artículo, precio y suma parcial.

El Cajero repite los pasos 3 a 4 hasta terminar los artículos del Cliente.

5. **Cajero**: Indica al TPV que se concluyó la captura de productos.
6. **Sistema**: Calcula y presenta el total con impuestos de la venta.
7. **Cajero**: Le indica el total de la venta al Cliente.
8. **Cliente**: Efectúa un pago.
9. **Cajero**: Gestiona el pago.
10. **Sistema**: Registra la venta. Genera un recibo.
11. **Cajero**: Da al Cliente el recibo impreso.
12. **Cliente**: Se marcha con los artículos comprados.

Ejemplo TPV: caso de uso completo (3)

Extensiones (o cursos alternativos):

Paso 3: Identificador incorrecto:

1. **Sistema**: Indica error y rechaza la entrada.

Pasos 3-7: a) El cliente le pide al Cajero que elimine un artículo de la compra:

1. **Cajero:** Introduce el identificador del artículo para eliminarlo.
2. **Sistema**: Registra la eliminación y muestra la suma parcial actualizada.

Pasos 3-7: b) El cliente le pide al Cajero que cancele la compra:

1. **Cajero:** Cancela la venta.
2. **Sistema**: Elimina los datos sobre la venta actual.

Pasos 8-9: a) Pago en efectivo:

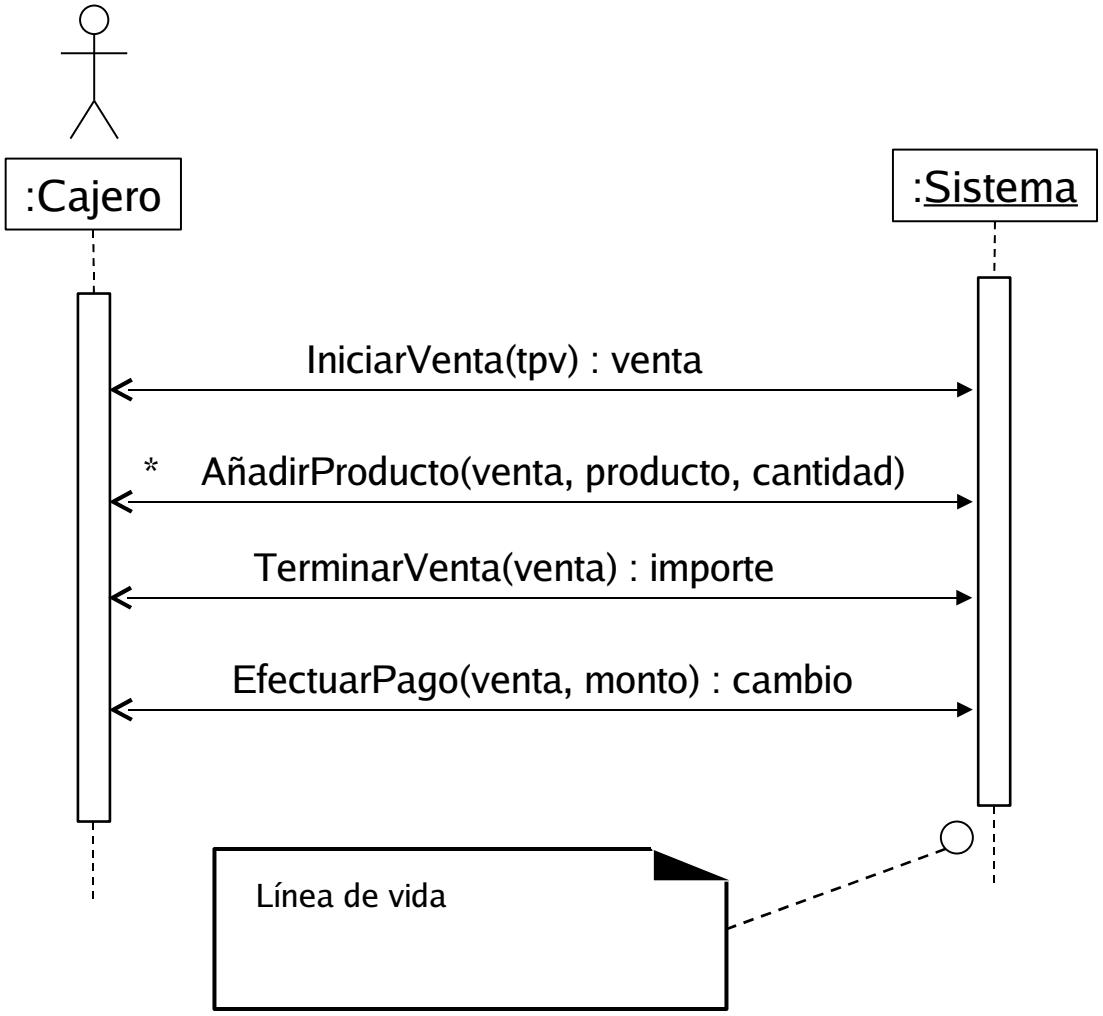
1. **Cliente**: Efectúa un pago en efectivo.
2. **Cajero**: Registra la cantidad de efectivo ofrecida.
3. **Sistema**: Muestra al Cajero la diferencia. Abre la caja.
4. **Cajero**: Da al Cliente el cambio y el recibo impreso. Cierra la caja.

Pasos 8-9: b) Pago con tarjeta:

1. **Cliente**: Entrega al Cajero la tarjeta de crédito ...

...

Ejemplo: Caso de uso Comprar productos



Contratos de las operaciones

- Describe el comportamiento del SI cuando se invoca una operación en términos de:
  - Cuáles son los cambios de estado de los datos
  - Cuáles son las salidas que el sistema proporciona
- Incluyen:
  - **Precondiciones** y **poscondiciones** que describen cambios de estado
  - **Salidas**
- Descripción declarativa:
  - **Qué** hará la operación más que **cómo** la hará.
- Establece un vínculo entre las operaciones y el esquema conceptual

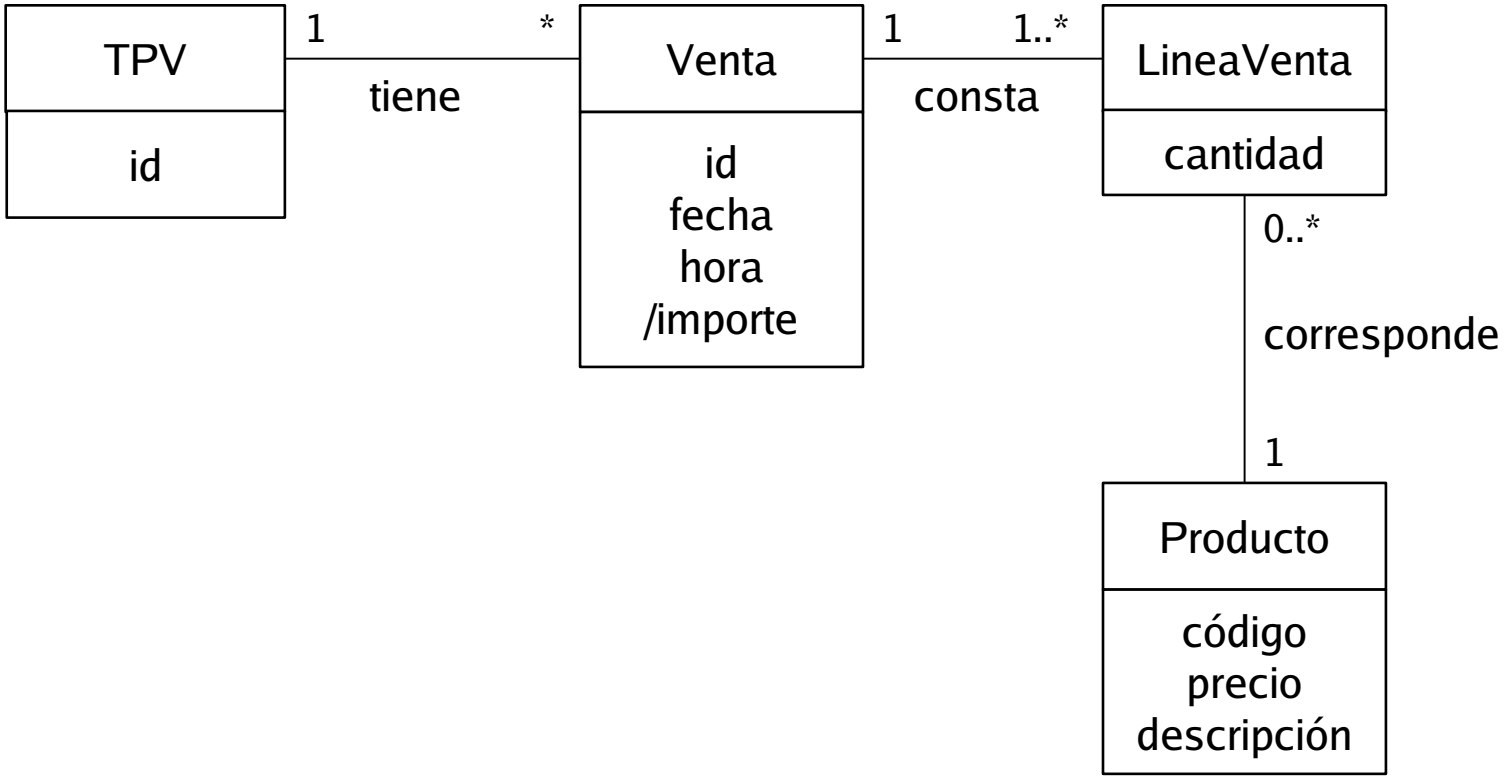
Componentes de los Contratos

- Name: nombre y argumentos de la operación (signatura)
- Responsabilities
  - Descripción informal del propósito de la operación
- Preconditions
  - Suposiciones sobre el estado del sistema antes de la operación
- Postconditions: cambios de estado que se han producido en el Modelo de Dominio
  - Altas/bajas de instancias de objetos
  - Altas/bajas de instancias de asociaciones
  - Modificación de atributos
- Salida
  - Descripción de la salida de la operación

Elaboración de Contratos

- Identificar las operaciones a partir del diagrama de secuencia
- Elaborar un contrato para cada operación
- Comenzar redactando la sección de Responsabilidades; después informalmente el propósito de la operación
- Completar la sección de Poscondiciones; describiendo de forma declarativa los cambios de estado del Modelo Conceptual
  - Creación y eliminación de objetos
  - Modificación de los atributos
  - Formación y cancelación de asociaciones

Ejemplo: contratos



Ejemplo: operación IniciarVenta

- **Name**:IniciarVenta(IdTPV) : IdVenta
- **Responsabilities**
  - Iniciar el registro de una venta
- **Preconditions**
  - Existe un TPV.id = IdTPV
- **Postconditions**
  - Se dio de alta una instancia v de Venta
  - v.id = un nuevo identificador para venta
  - v.fecha = fecha actual del sistema
  - h.hora = hora actual del sistema
  - Se dio de alta una instancia de la asociación 'tiene' entre v y la instancia de TPV.id = IdTPV
- **Salida**
  - v.id

Ejemplo: operación AñadirProducto

- **Name**:AñadirProducto(IdVenta, código, c) : descripción, precio, importe
- **Responsabilities**
  - Registrar una línea de venta
- **Preconditions**
  - Existe una venta v con v.id = IdVenta
  - Existe un producto p con p.código = código
- **Postconditions**
  - Se dió de alta una instancia de LíneaVenta lv
  - lv.cantidad = c
  - Se dió de alta una instancia de la asociación 'consta' entre lv y v
  - Se dió de alta una instancia de la asociación 'corresponde' que asocia lv y el producto p con p.código = código
  - v.importe = v.importe + lv.cantidad * p.precio
- **Salida**
  - p.descripción, p.precio, v.importe

Ejemplo: operación TerminarVenta

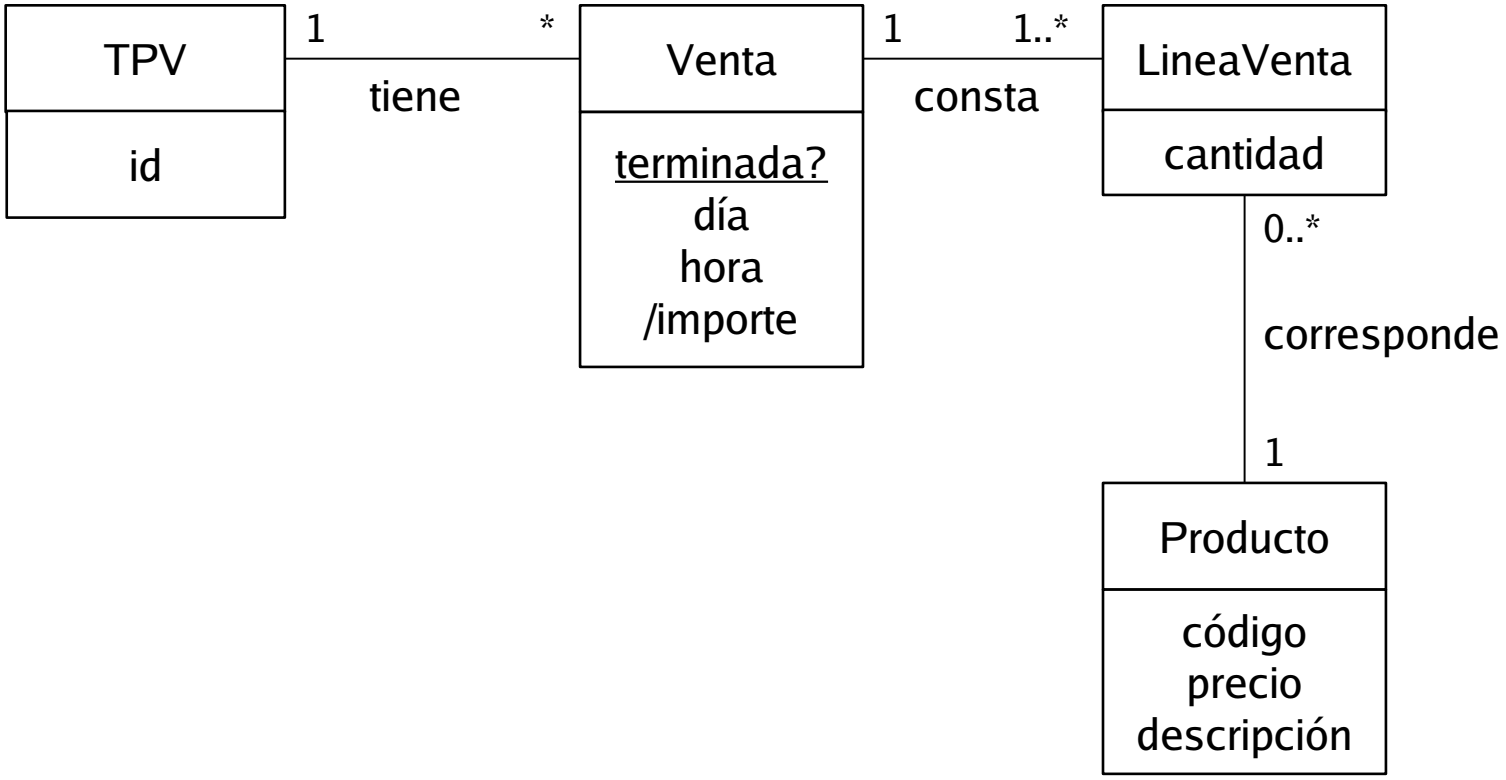
- **Name**:TerminarVenta(IdVenta) : importe
- **Responsabilities**
  - Finalizar el registro de una venta y mostrar el importe a de la venta
- **Preconditions**
  - Existe una venta v con v.id = IdVenta
- **Postconditions**
  - v.terminada? = verdadero
- **Salida**
  - importe = v.importe

Ejemplo: operación EfectuarPago

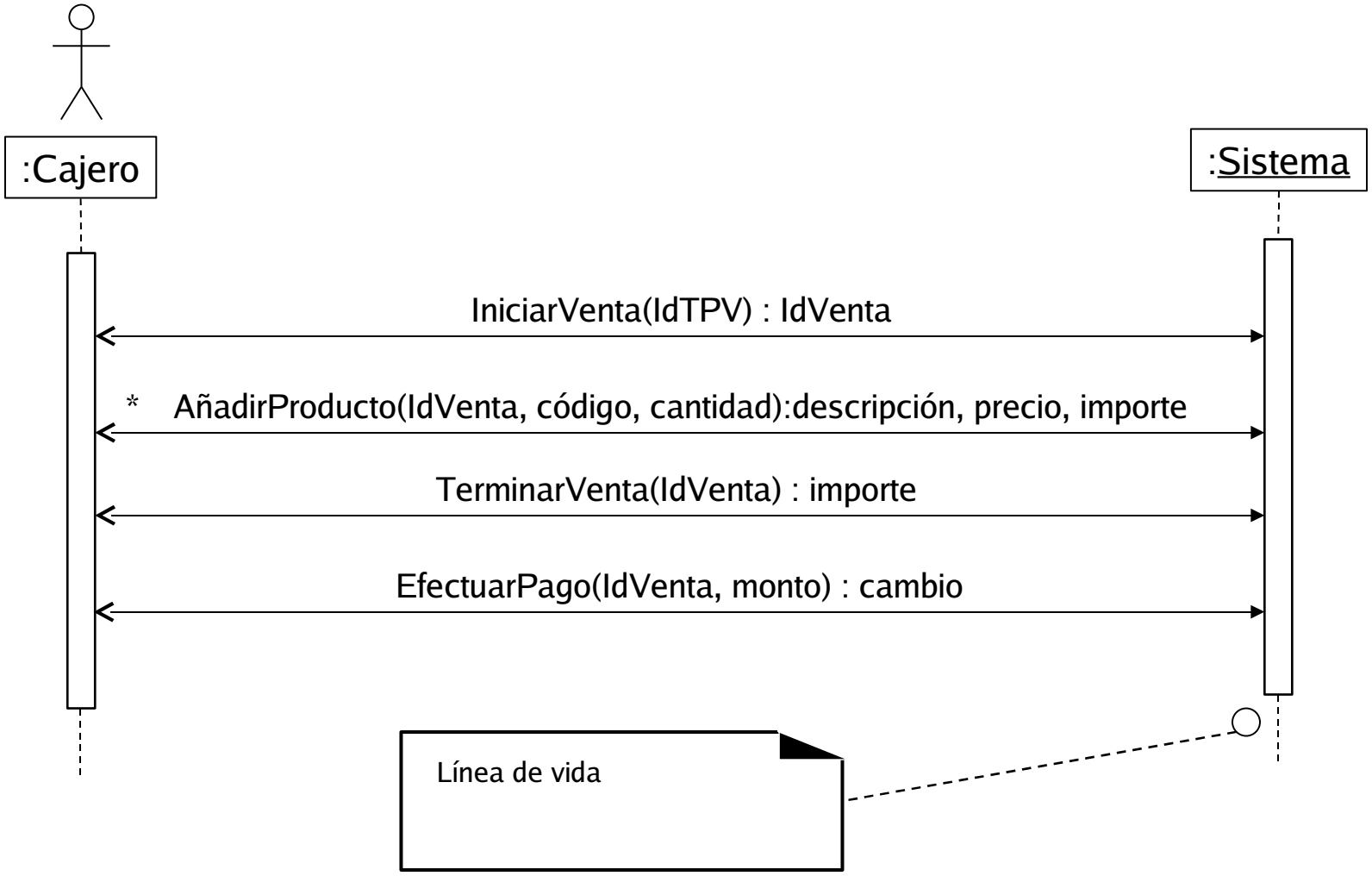
- **Name**:EfectuarPago(IdVenta, monto) : cambio
- **Responsabilities**
  - Mostrar el cambio a devolver
- **Preconditions**
  - Existe una venta v con v.id = IdVenta
  - v.terminada? = verdadero (la venta está concluida)
  - monto >= v.importe
- **Postconditions**

- **Salida**
  - cambio = monto - v.importe

Ejemplo: contratos



Ejemplo: Caso de uso Comprar productos



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 prestamo 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)

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 prestamo.
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 prestamo.

Ejemplo: Caso de uso Completo (2)

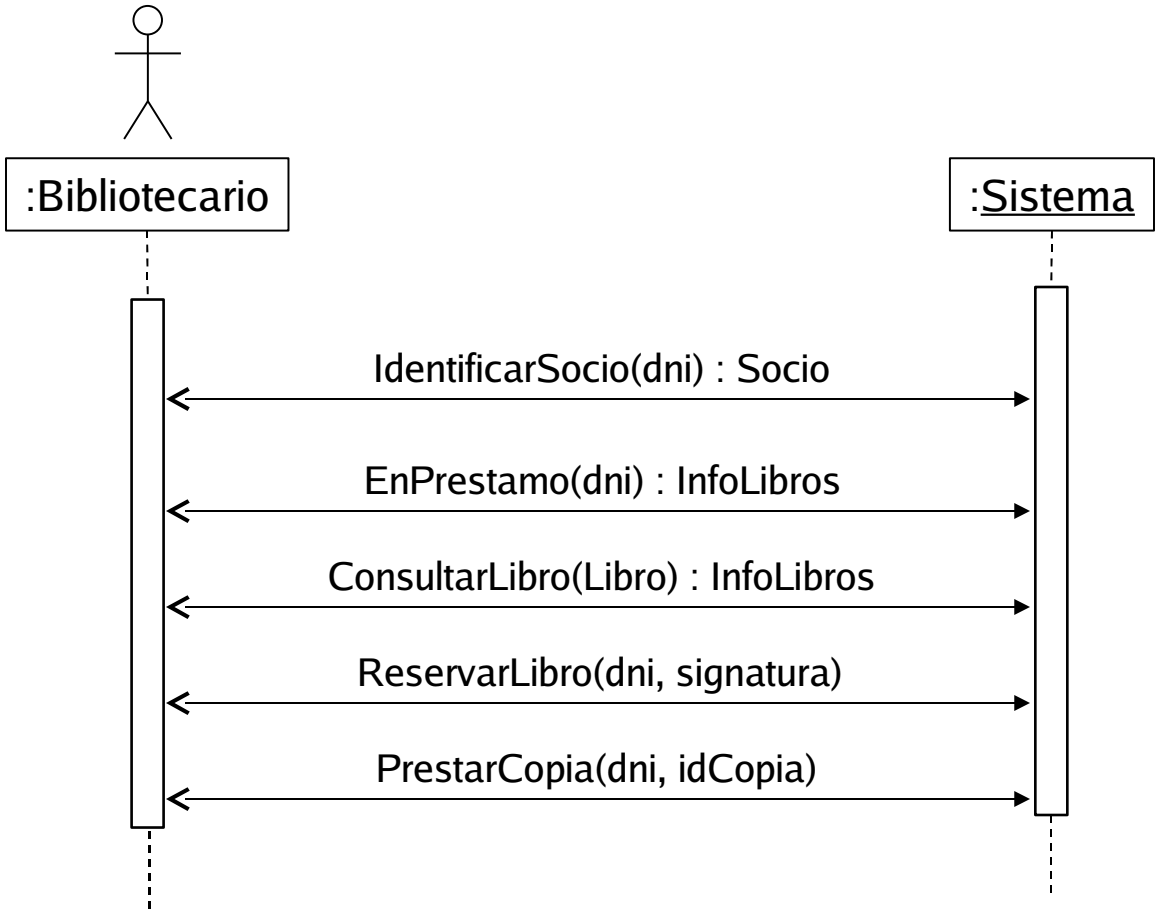
Extensiones (o cursos alternativos)
Paso 2: USES Identificar Socio.
Paso 4 a): El Socio ya tiene el máximo de libros prestados.

1. **Bibliotecario**: Sugiere devolver algún libro.

Paso 4 b):
EXTENDS Consultar catálogo.
Paso 6-7: Todas las copias del libro ya estan prestadas.
EXTENDS Reservar libro.
Paso 7: El Socio no puede confirmar el libro o no acepta la fecha de devolución.

1. **Bibliotecario**: Cancela el prestamo en curso.

Ejemplo biblioteca: Pedir Libro



Ejemplo Biblioteca: operación EnPrestamo

- **Name**:EnPrestamo(dni) : InfoLibros
- **Responsabilities**

  Mostrar los copias prestadas al socio. La información incluye el periodo de préstamo y la fecha de devolución de cada copia
- **Preconditions**

  Existe un socio s.dni = dni
- **Postconditions**
- **Salida**

  Todas las copias

  c = (Copia.prestadaA -> select(s.dni = dni))

  Todas sus fechas de inicio y fin de préstamo

  p = (c.préstamoC)

Ejemplo Biblioteca: operación PrestarCopia

- **Name**:PrestarCopia(dni, signatura)
- **Responsabilities**

  Registra el nuevo préstamo con la fecha actual.
- **Preconditions**

  Existe un socio s.dni = dni

  Existe una copia c.idCopia = idCopia
- **Postconditions**

  Se creó una instancia pc de la asociación PréstamoC

  Se asoció pc a la cópia c y al sócio s

  pc.iniFecha = fechaActual()

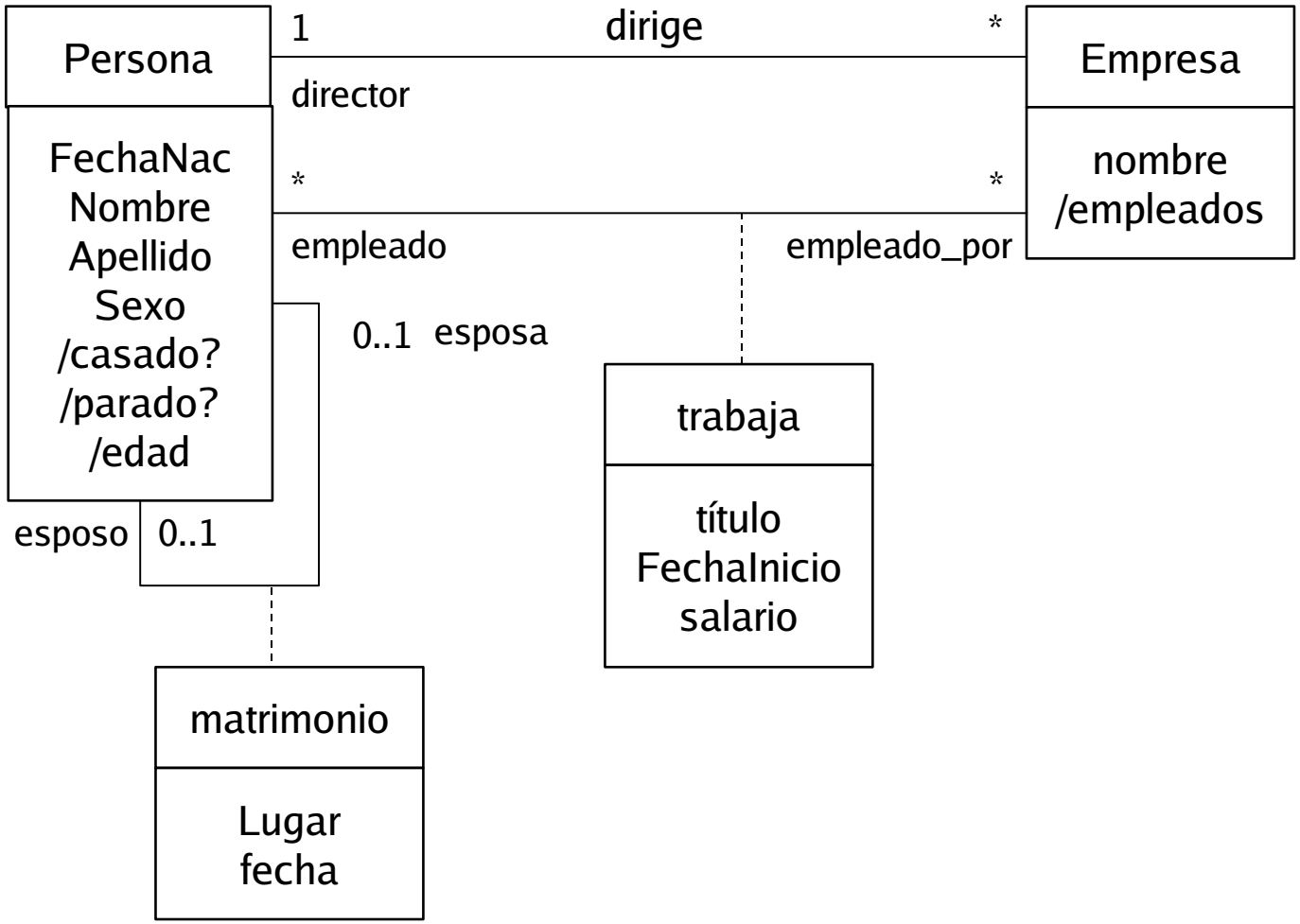
  pc.finFecha = fechaActual() + c.tiempoMax

  c.estado = prestado
- **Salida**

Object Constraint Language (OCL)

- Los modelos gráficos no son suficientes para una especificación precisa y no ambigua
- OCL
  - Lenguaje formal
  - Permite definir expresiones
  - No es un lenguaje de programación!
  - Permite especificar invariantes (restricciones y condiciones)
  - Permite navegar entre los objetos

Ejemplo: OCL


Persona
FechaNac
Nombre
Apellido
Sexo
/casado?
/parado?
/edad
1
dirige
*
director
Empresa
nombre
/empleados
*
*
empleado
empleado_por
0..1
esposa
esposo
0..1
trabaja
título
FechaInicio
salario
matrimonio
Lugar
fecha

Expresiones OCL

- Una expresión OCL describe propiedades de los objetos del Modelo Conceptual y
- Una propiedad puede hacer referencia a:
  - Atributos de una clase de objetos

    Restricción de Integridad

    "Las personas tienen edades superiores o iguales a cero"

    Persona — self.edad >= 0

    p:Persona — p.edad >= 0 -- *instancia textual*
  - Navegación a través de las asociaciones

    Empresa

    self.director – Persona

    self.director.nombre – string

Expresiones OCL: colecciones

- Una colección de elementos puede ser del tipo
  - Conjunto: no hay elementos repetidos
  - Bolsa (multiconjunto): puede haber elementos repetidos
  - Secuencia: bolsa ordenada

- Reglas de navegación
  - Si la multiplicidad de la asociación es 1, el resultado es un objeto o un conjunto con un único objeto
  - Si la multiplicidad de la asociación es >1, el resultado es una bolsa (o, a veces, un conjunto)

Expresiones OCL: operaciones sobre colecciones (1)

- **Select**: selecciona un subconjunto de la colección

  "Personas mayores de 50 años que trabajen en una empresa"

  Empresa

  ```
  self.empleado -> select(edad>50)
  self.empleado -> select(p:Persona, p.edad >50)
  ```

- **Collect**: selecciona una colección que deriva de otra

  "edades (con repetidos) de los empleados de una empresa"

  Empresa

  ```
  self.empleado -> collect(FechaNac)
  ```

Expresiones OCL: operaciones sobre colecciones (2)

- **forAll**: expresión que deben satisfacer todos los elementos

  "Todos los empleados de la empresa tienen menos de 65 años"

  Empresa

  ```
  self.empleado -> forAll(edad<65)
  ```

- **Exists**: condición que satisface un elemento

  "Alguien de la empresa tiene menos de 30 años"

  Empresa

  ```
  self.empleado -> exists(edad<30)
  ```

Expresiones OCL: operaciones complejas

- "Las personas casadas deben ser mayores de edad"

  Persona

  self.esposa -> notEmpty implies self.esposa.edad >= 18 and
  self.esposo -> notEmpty implies self.esposo.edad >= 18

- "Una empresa tiene como máximo 50 empleados"

  Empresa

  self.empleado -> size <= 50

- "Definición del atributo derivado /empleados"

  Empresa

  self.empleados = (self.empleado -> size)

- "Definición del atributo derivado parado"

  Persona

  self.parado?= if self.empleado_por isEmpty then true else false

Expresiones OCL: navegación por clases asociativas

- Navegación a una clase asociativa

  "Los sueldos de las personas que trabajan en la UPV deben ser mayores a 100.000 euros"

  Persona

  (self.empleado_por -> select(nombre='UPV')).trabaja ->

  forAll(salario > 100.000)

- Navegación desde una clase asociativa

  "Las personas que trabajan no pueden estar en paro"

  Trabaja

  self.empleado.parado? = false

  self.persona.parado? = false

  si no existe rol en el extremo de la asociación

Expresiones OCL: expresiones sinónimas

- Las expresiones empiezan siempre en la instancia contextual
- Una misma expresión puede ser especificada de formas distintas

  Persona

  ```
  self.esposa -> notEmpty implies self.esposa.edad >= 18 and
  self.esposo -> notEmpty implies self.esposo.edad >= 18
  ```

  Matrimonio

  ```
  self.esposa.edat >= 18 and self.esposo.edat >= 18
  ```

- Indicadores para escoger la instancia contextual
  - Clase del atributo que queremos restringir
  - Si se desean restringir atributos de varias clases, cualquiera
  - Las restricciones deben navegar por el menor número de asociaciones

Expresiones OCL: ejemplos

- "Todos los trabajadores deben ser mayores de edad"

  Persona

  self.edad >= 18 – *es preferible a*

  Empresa

  self.empleado -> forAll(edad > 18)

- "No se permiten matrimonios entre los trabajadores"

  Empresa

  self.empleado.esposo -> intersection(self.empleado) -> isEmpty

  – *es preferible a*

  Empresa

  self.empleado.esposo -> intersection(self.empleado) -> isEmpty

  and

  self.empleado.esposa -> intersection(self.empleado) -> isEmpty

Expresiones OCL: Operaciones estándar booleanas

Operación	Notación	Resultado
or	a or b	booleano
and	a and b	booleano
or exclusivo	a xor b	booleano
negación	not a	booleano
igualdad	a = b	booleano
desigualdad	a <> b	booleano
implicación	a implies b	booleano
if-then-else	if c then b else b'	b or b'

Expresiones OCL: Operaciones estándar string

Operación	Notación	Resultado
concatenación	string.concat(string)	string
tamaño	string.size	entero
substring	string.substring(int,int)	string
igualdad	string1 = string2	booleano
desigualdad	string1 <> string2	booleano

Expresiones OCL: Operaciones estándar colecciones

Operación	Resultado
size	número de elementos
count(object)	número de las ocurrencias del objeto
includes(object)	el objeto pertenece a la colección?
isEmpty	la colección está vacía?
notEmpty	la colección no está vacía?
sum()	suma de todos los elementos
exists(expression)	*expression* es cierta para algún elemento?
forAll(expression)	*expression* es cierta para todos los elementos?
select(expression)	selecciona los elementos para los que *expression* es cierta
reject(expression)	elimina los elementos para los que es falsa
union(collection)	la unión de las dos colecciones
intersection(collection)	la intersección de las dos colecciones