Construcción de Software

Capítulo 1

El Lenguaje Unificado de Modelado, UML

Capítulo 1. Estructura

- Presentación de UML
- Necesidad del modelado
- Modelado de casos de Uso
  - Diagrama de casos de uso
- Modelado estructural
  - Diagrama de clases
- Paquetes
- Vistas de UML
- Modelado dinámico
  - Diagramas de interacción

Capítulo 1. Estructura (II)

- Modelado de flujos de trabajo
  - Diagramas de actividades
- Modelado del estado
  - Máquinas de estado
- Modelado de la implementación
  - Diagramas de componentes
  - Diagramas de despliegue
- Colaboraciones
- OCL (*Object Constraint Language)*

Capítulo 1. Bibliografía

- [Booch et al. 99] Booch, G. et al. *"El lenguaje unificado de modelado*", Addison-Wesley, 1999.
- [Larman 02] Larman, C. *"UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado"*, Segunda Edición, Prentice-Hall, 2002.

El lenguaje unificado de modelado, UML

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

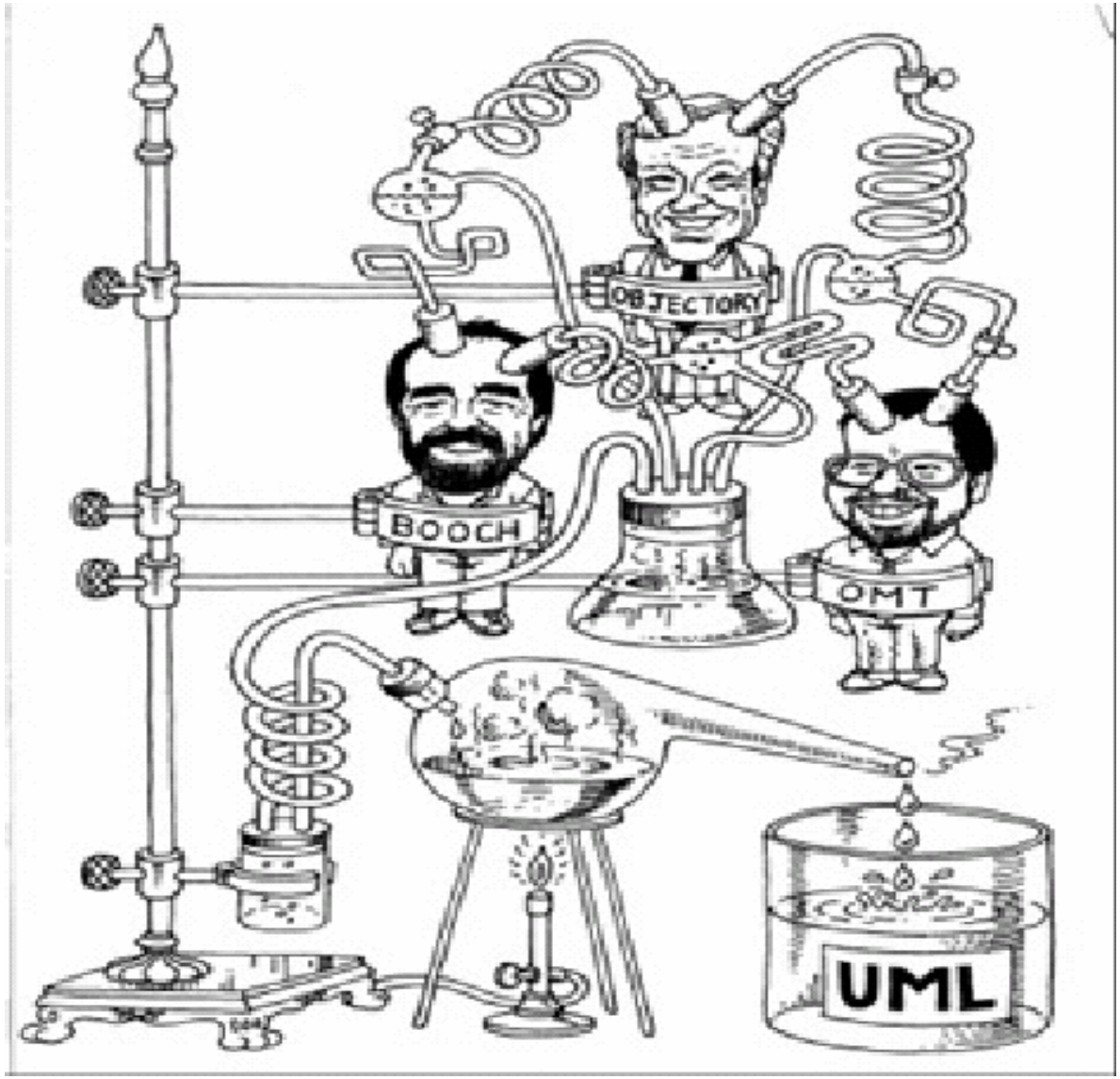
  Unified **M**odeling **L**anguage (UML)
- Proceso de estandarización promovido por el OMG

El consorcio OMG

- Rational Software
- Oracle
- IBM
- DEC
- Microsoft
- Hewlett-Packard
- Sterling Software
- MCI Systemhouse
- Unisys
- IntelliCorp
- ICON Computing
- i-Logix
- ObjectTime
- Platinum Technology
- Petch
- Taskon A/S
- Reich Technologies
- Softeam

....

Cómo se creó UML

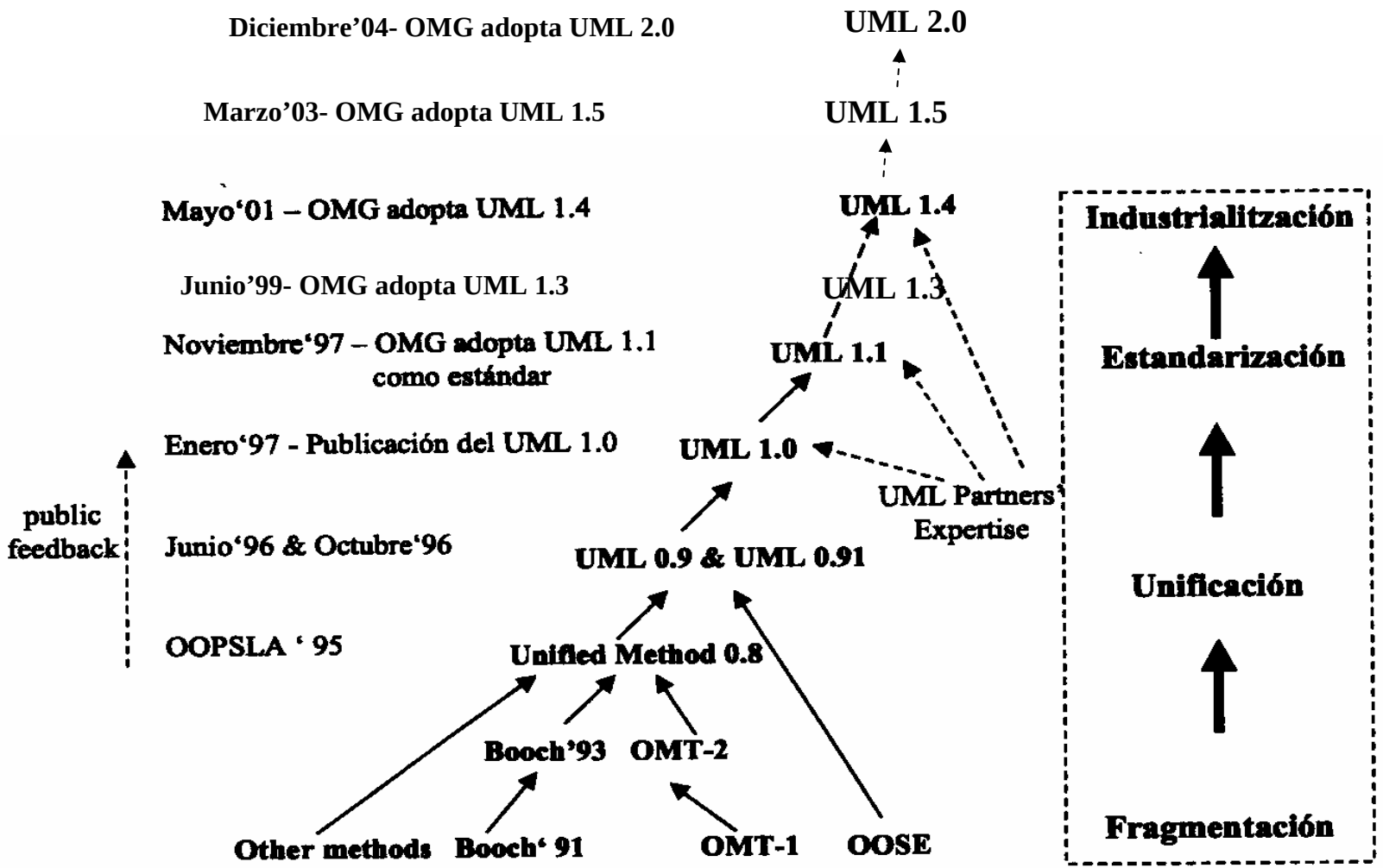


De “Introduction to the Unified Modeling Language”, Terry Quatrani, UML, http://www.rational.com/media/uml/intro_rdn.pdf

Las raíces técnicas de UML

- OMT - *Object Modeling Technique* (Rumbaugh et al.)
  - especialmente bueno para análisis de datos de SI
  - entre otros, usa extensiones de los diagramas ER
- Método-Booch (G. Booch)
  - especialmente útil para sistemas concurrentes y de tiempo real
  - fuerte relación con lenguajes de programación, como Ada
- OOSE - *Object-Oriented Software Engineering* (I. Jacobson)
  - desarrollo guiado por los *use cases*
  - buen soporte de Ingeniería de Requisitos e Ingeniería de Información
  - Modelado y simulación de sistemas de telecomunicaciones
- UML unifica estos conceptos e introduce otros nuevos

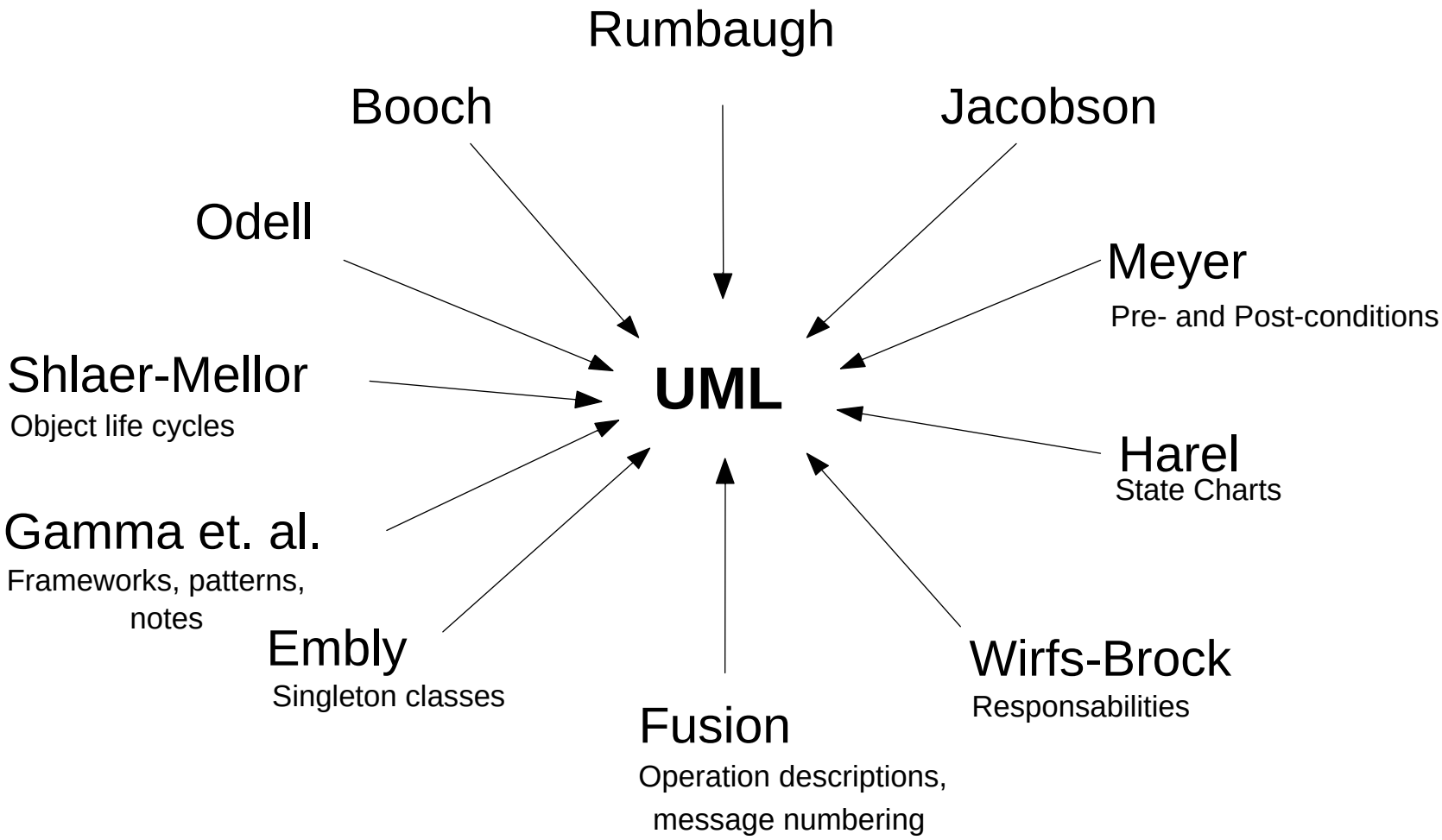
Evolución de UML



Los rivales de UML

- Otras propuestas enviadas a OMG
  - Catalysis (D. D'Souza, A. Willis)
  - Syntropy (S. Cook et al., IBM)
  - OML/Open (B. Henderson-Sellers)
  - Fusion (D. Coleman, HP)

UML aglutina otros enfoques



(Tomada de www.dsic.upv.es/~uml)

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.
- Cubrir las cuestiones relacionadas con el tamaño propias de los sistemas complejos y críticos.
- Lenguaje utilizable por las personas y las máquinas
- Encontrar equilibrio entre expresividad y simplicidad.

UML y el 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 están definiendo muchos procesos para UML.
- Rational ha ideado *RUP*, *"Proceso Unificado de Rational"*.
- Utilizable para sistemas que no sean software

¿Por qué modelamos?

"Una empresa software con éxito es aquella que produce de manera consistente software de calidad que satisface las necesidades de los usuarios"

"El modelado es la parte esencial de todas las actividades que conducen a la producción de software de calidad"

¿Por qué modelamos?

"Un modelo es una simplificación de la realidad"

- Construimos modelos para comprender mejor el sistema que estamos desarrollando
- Cuatro utilidades de los modelos:
  - Visualizar cómo es o queremos que sea el sistema
  - Especificar la estructura y comportamiento del sistema
  - Proporcionan plantillas que guían la construcción del sistema
  - Documentan las decisiones
- "Equivalen a los ***planos*** de un edificio"

Principios del modelado

- La elección de los modelos tiene una profunda influencia sobre cómo se acomete el problema y se moldea la solución.
- Todo modelo puede expresarse a diferentes niveles de detalle y usarse en diferentes momentos del ciclo de vida.
- Todo modelo debe estar ligado a la realidad.
- Un único modelo no es suficiente. Cualquier sistema no trivial se aborda mejor a través de un pequeño conjunto de modelos casi independientes, que muestran distintos aspectos.

¿Por qué las empresas no hacen modelado?

- La mayor parte de las empresas software no realizan ningún modelado.
- El modelado requiere:
  - aplicar un proceso de desarrollo
  - formación del equipo en la técnicas
  - tiempo
- ¿Se obtienen beneficios con el modelado?

¿Problemas en el desarrollo de software?

- Retrasos en los plazos
- Proyectos cancelados
- Rápido deterioro del sistema instalado
- Tasa de defectos o fallos
- Requisitos mal comprendidos
- Cambios frecuentes en el dominio del problema
- Muchas de las interesantes características del software no proporcionan beneficios al cliente

¿Modelado es la solución?

Utilidad del modelado

¿Escribimos código directamente?

- Se facilita la comunicación entre el equipo al existir un lenguaje común.
- Se dispone de documentación que trasciende al proyecto.
- Hay estructuras que trascienden lo representable en un lenguaje de programación.

Utilidad de UML

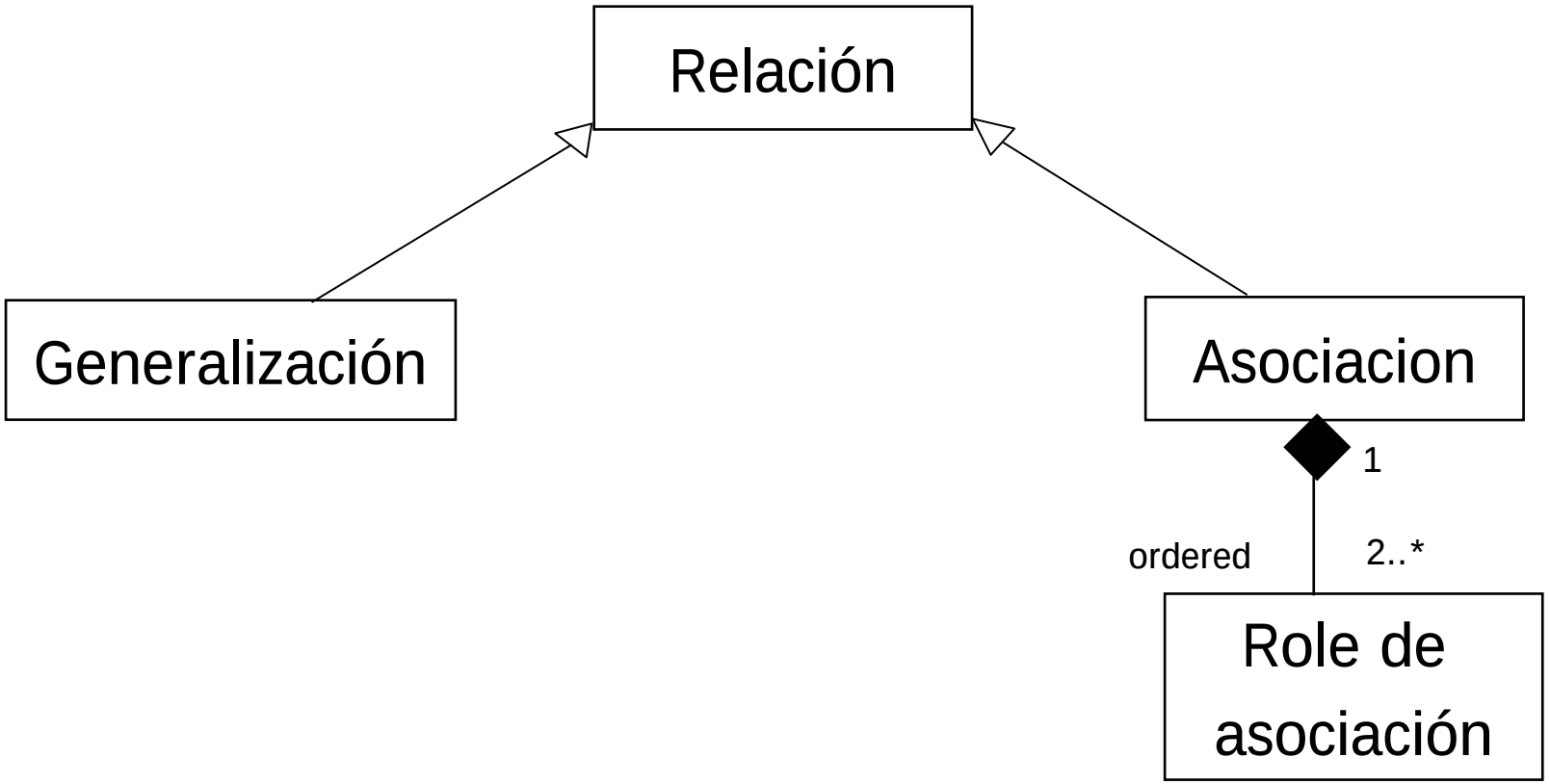
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose modelos precisos, no ambiguos y completos.
- UML puede conectarse a lenguajes de programación:
  - Ingeniería directa e inversa
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones,..)

Metamodelo UML

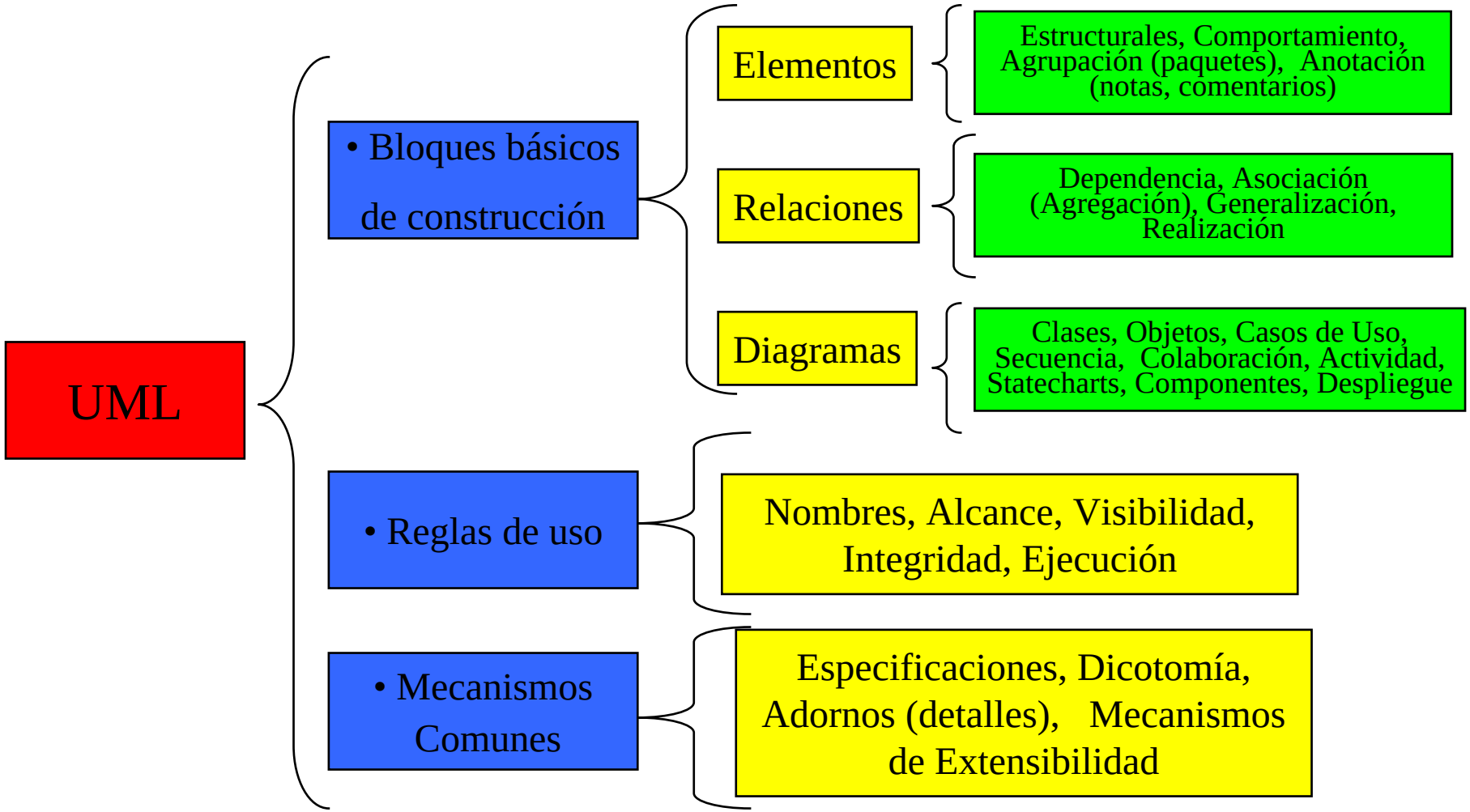
- ¿Cómo se expresa la semántica del modelo?
  - Informalmente
  - Formalmente

- El metamodelo UML define la notación de un modo riguroso, a través de diagramas de la propia notación y con OCL *(Object Constraint Language)*.

Metamodelo UML: Ejemplo



Cómo se organiza UML

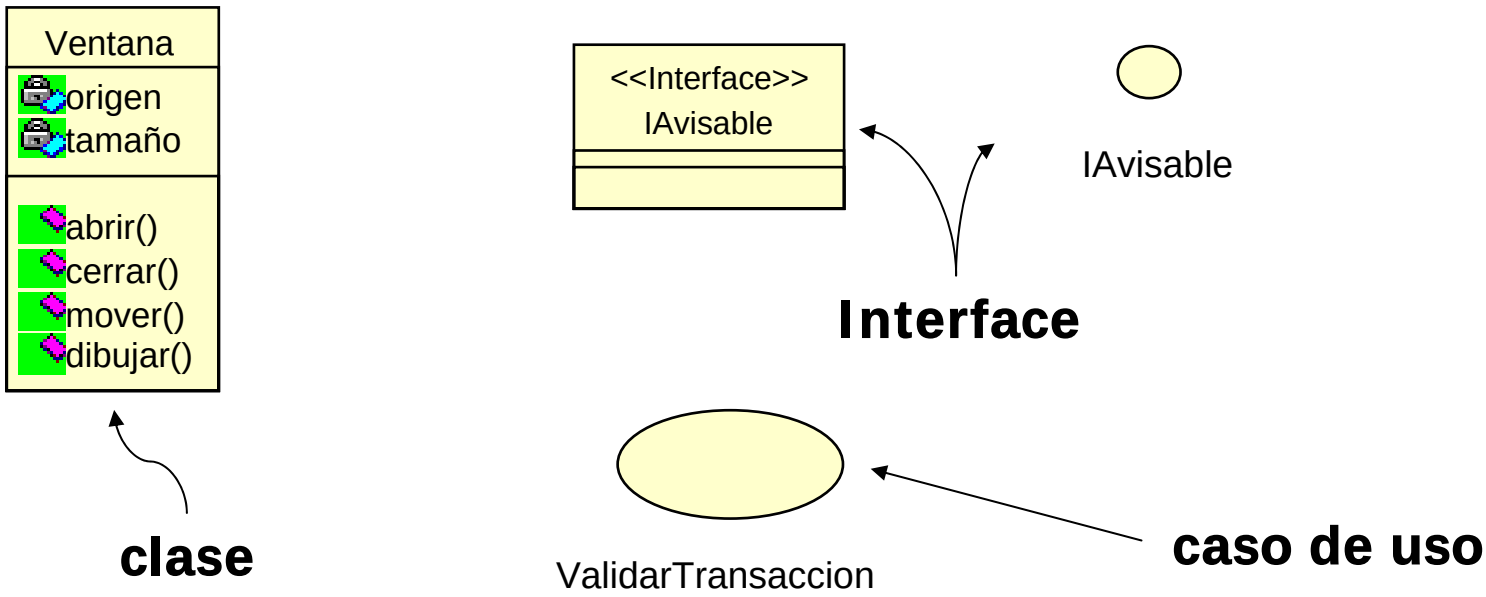


Modelo Conceptual de UML

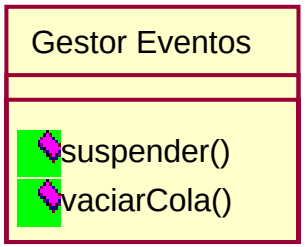
- Bloques básicos de construcción
  - Elementos
    - Estructurales, Comportamiento, Agrupación, Anotación
  - Relaciones
  - Diagramas
- Reglas para combinar bloques
  - Establecen qué es un modelo bien formado
- Mecanismos comunes

Elementos Estructurales

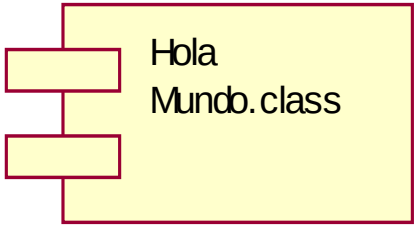
- Partes estáticas de un modelo



Elementos Estructurales

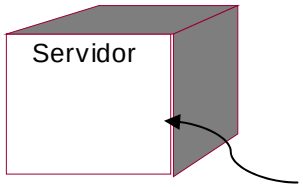


clase activa (UML 1.X)



componente (UML 1.X)

colaboración



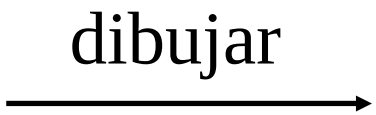
nodo

Elementos de Comportamiento

Son las partes dinámicas de UML.

Interacción

Conjunto de mensajes intercambiados entre un conjunto de objetos con un propósito particular.



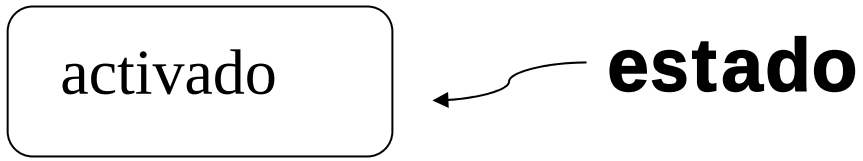
mensaje

Elementos de Comportamiento

Son las partes dinámicas de UML.

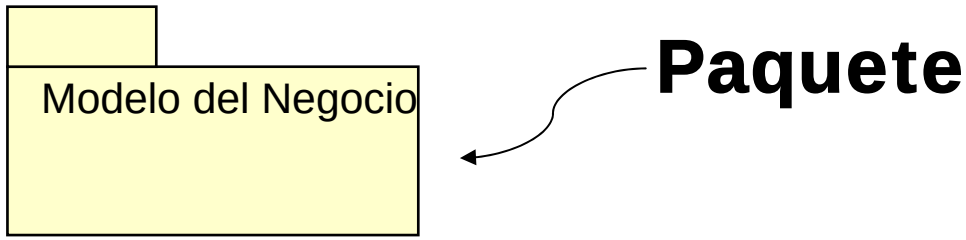
Máquina de estados

Secuencia de estados por las que pasa un objeto durante su vida en respuesta a eventos.



Elementos de Agrupación

Son las partes organizativas de los modelos UML

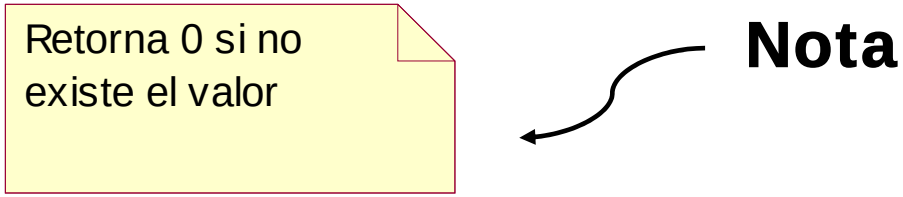


Un paquete incluye un conjunto de elementos de cualquier naturaleza.

Tiene una naturaleza conceptual.

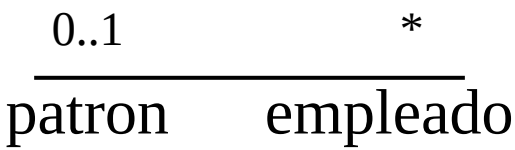
Elementos de Anotación

Son las partes explicativas de los modelos UML

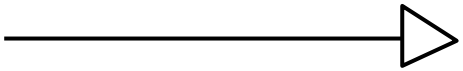


Relaciones

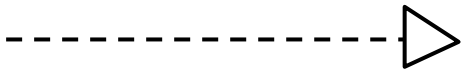
Dependencias



Asociaciones



Generalizaciones



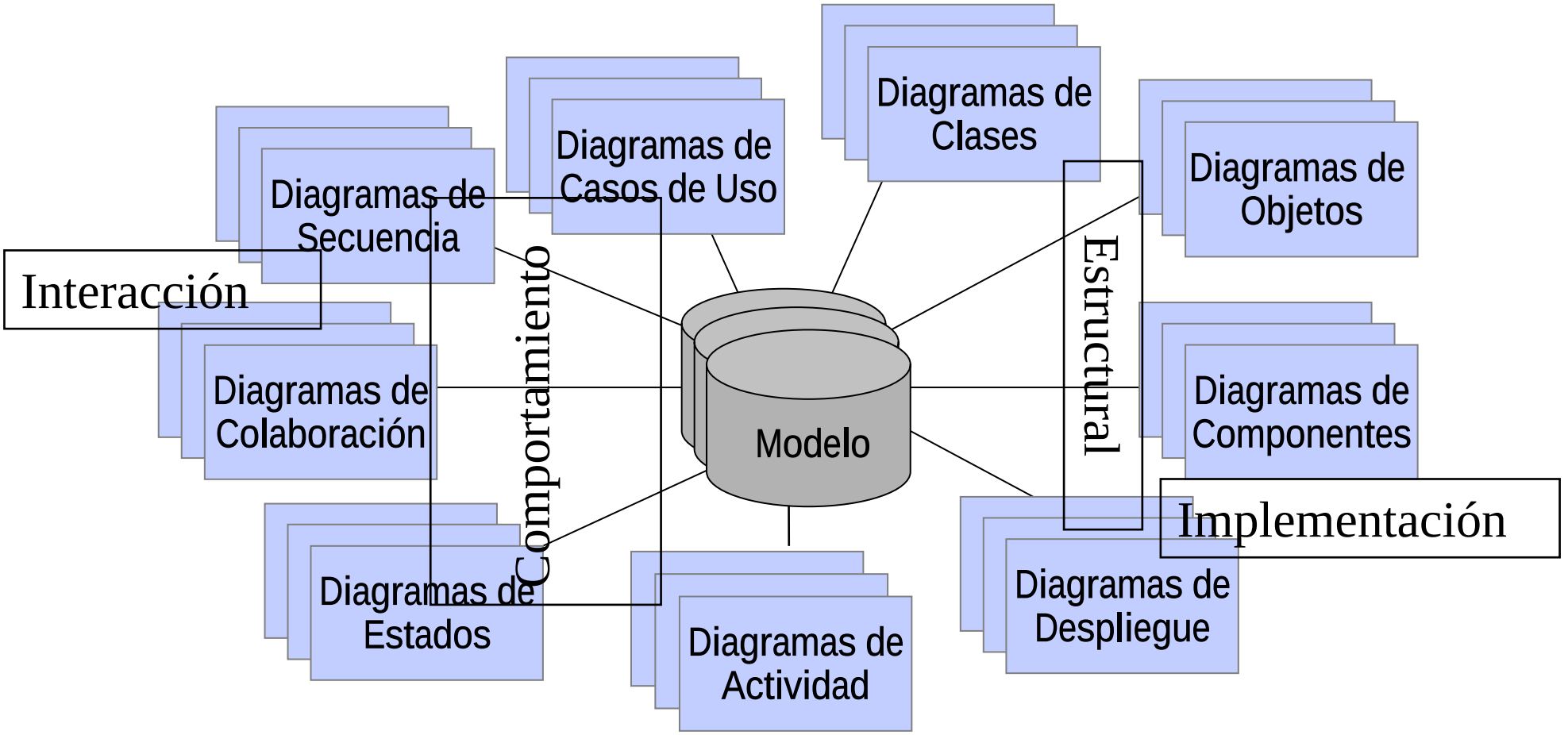
Realización

Diagramas de UML

- **Diagramas de Casos de Uso**
- **Diagramas de Clases**
- **Diagramas de Objetos**
- **Diagramas de Interacción**
  - Diagrama de Secuencia
  - Diagrama de Colaboración (*Comunicación* en UML 2.0)
- **Diagramas de Estados**
- **Diagramas de Actividades**
- **Diagramas de Componentes**
- **Diagramas de Despliegue**
- **Composite Structure Diagram (UML 2.0)**
- **Package Diagram (UML 2.0)**
- **Interaction Overview Diagram (UML 2.0)**
- **Timing Diagram (UML 2.0)**

http://www.agilemodeling.com/essays/umlDiagrams.htm

Diagramas de UML



Reglas de uso de UML

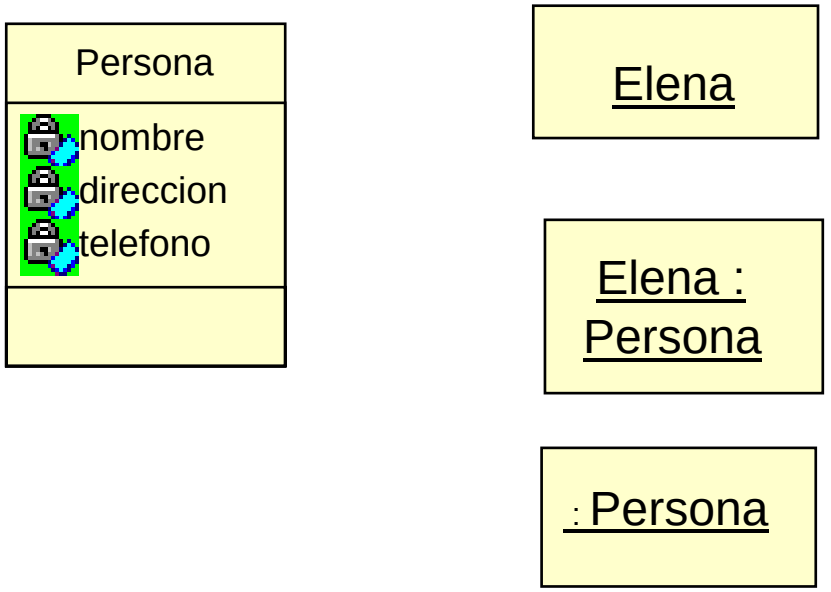
- Especifican cómo construir modelos bien formados (auto consistentes y en armonía con otros modelos relacionados)
- Proporcionan reglas semánticas para:
  - **Nombres** (a qué se puede llamar elementos, relaciones y diagramas)
  - **Alcance** (contexto que da significado específico a un nombre)
  - **Visibilidad** (cómo los nombres pueden ser vistos y usados por otros)
  - **Integridad** (cómo los elementos se relacionan entre sí de forma consistente)
  - **Ejecución** (significado de simular o ejecutar un modelo dinámico)
- Puede ser necesario trabajar con modelos “mal formados”:
  - Con omisiones (incluso intencionadamente, para simplificar las vistas)
  - Incompletos (faltan aún elementos, relaciones por identificar)
  - Inconsistentes (no se garantiza la integridad del modelo)

Mecanismos comunes de UML

- Especificaciones
  - Explicación textual de la sintaxis y la semántica de un bloque de construcción
  - Proporcionan una base semántica para cada elemento
  - Los diagramas son proyecciones (o vistas parciales) de esa base
- Adornos
  - La notación gráfica básica de cada elemento puede incluir adornos textuales o gráficos para resaltar algunas propiedades de la especificación.

Mecanismos comunes de UML

- Divisiones Comunes
  - Dicotomía clasificador /instancia



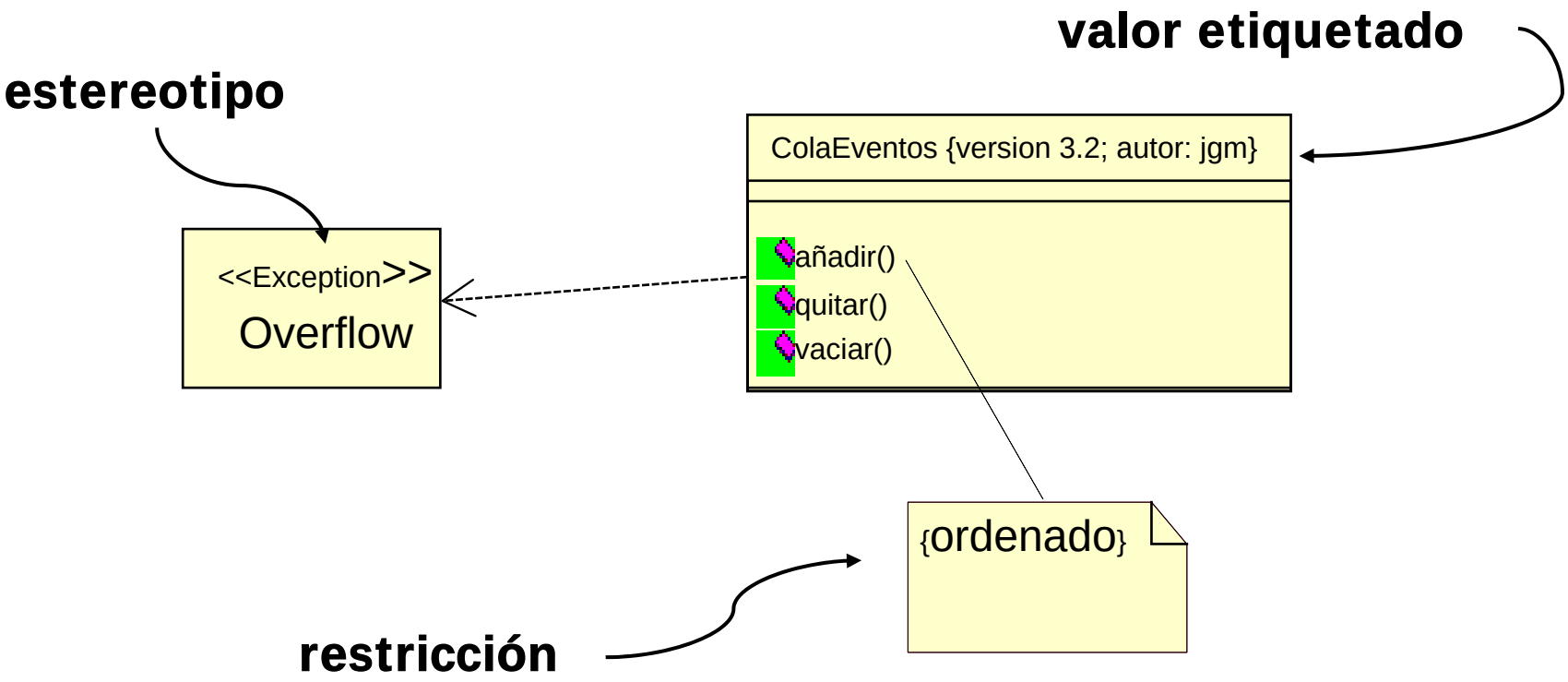
Mecanismos comunes de UML

- Divisiones Comunes
  - Dicotomía interfaz / implementación

Mecanismos comunes de UML

- Mecanismos de extensibilidad
  - **Estereotipos**
    - Extienden el vocabulario de UML, permitiendo añadir nuevos tipos de bloques de construcción
  - **Valores etiquetados**
    - Extienden las propiedades de un bloque de construcción, añadiendo nueva información
  - **Restricciones**
    - Extiende la semántica de un bloque, añadiendo reglas o modificando las existentes.

Mecanismos de extensibilidad de UML



Perfiles de UML

- "***Profiles***" (perfiles) de UML
  - Conjuntos pre definidos de estereotipos, valores etiquetados, restricciones e iconos de la notación que juntos especializan y adaptan UML a un dominio (modelado de negocios) o procesos específicos (como *Rational Unified Process*)

Modelado de Casos de Uso

- Un caso de uso especifica el comportamiento deseado del sistema.
- Representan los **requisitos funcionales** del sistema.

  "Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede ejecutar y que produce un resultado observable de valor para un particular actor"

- Describen *qué* hace el sistema, no *cómo* lo hace.

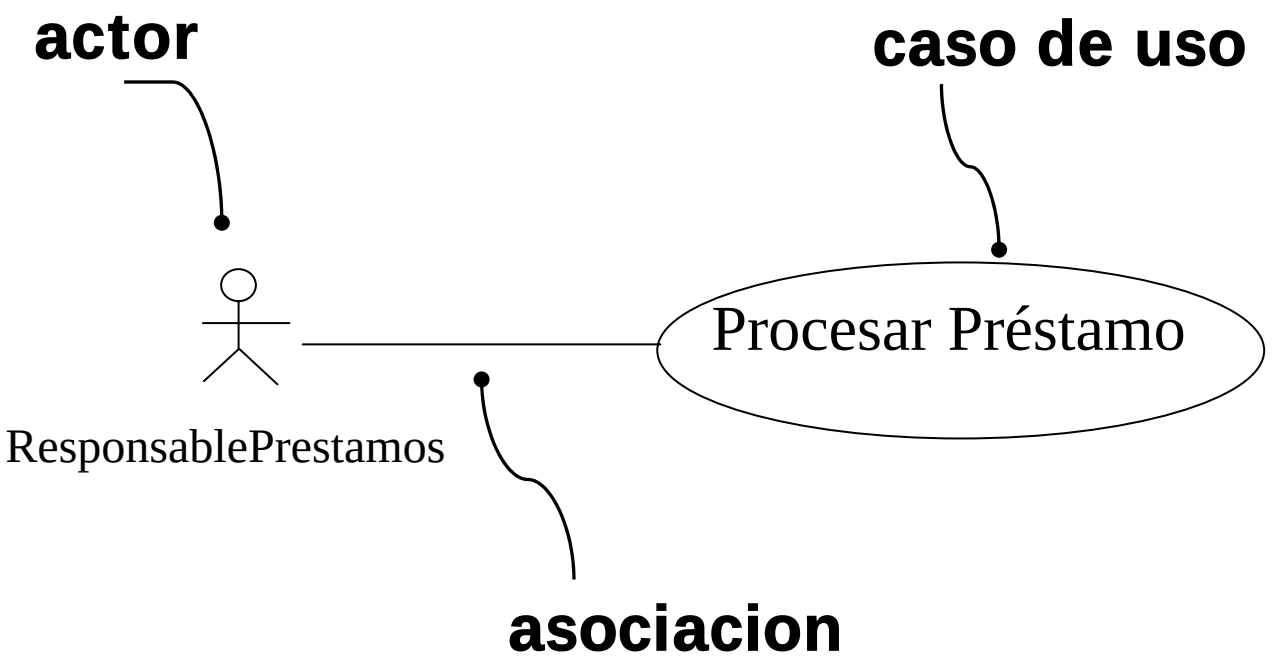
Modelado de Casos de uso

- Técnica de recolección y especificación de requisitos.
- Fáciles de comprender/validar por los usuarios.
- Guían todo el proceso de desarrollo.
- Ayudan a la planificación/desarrollo incremental.
- Tradicionalmente ligados a la OO
  - pero no obligatorio
- Ayudan a determinar la interfaz de usuario.

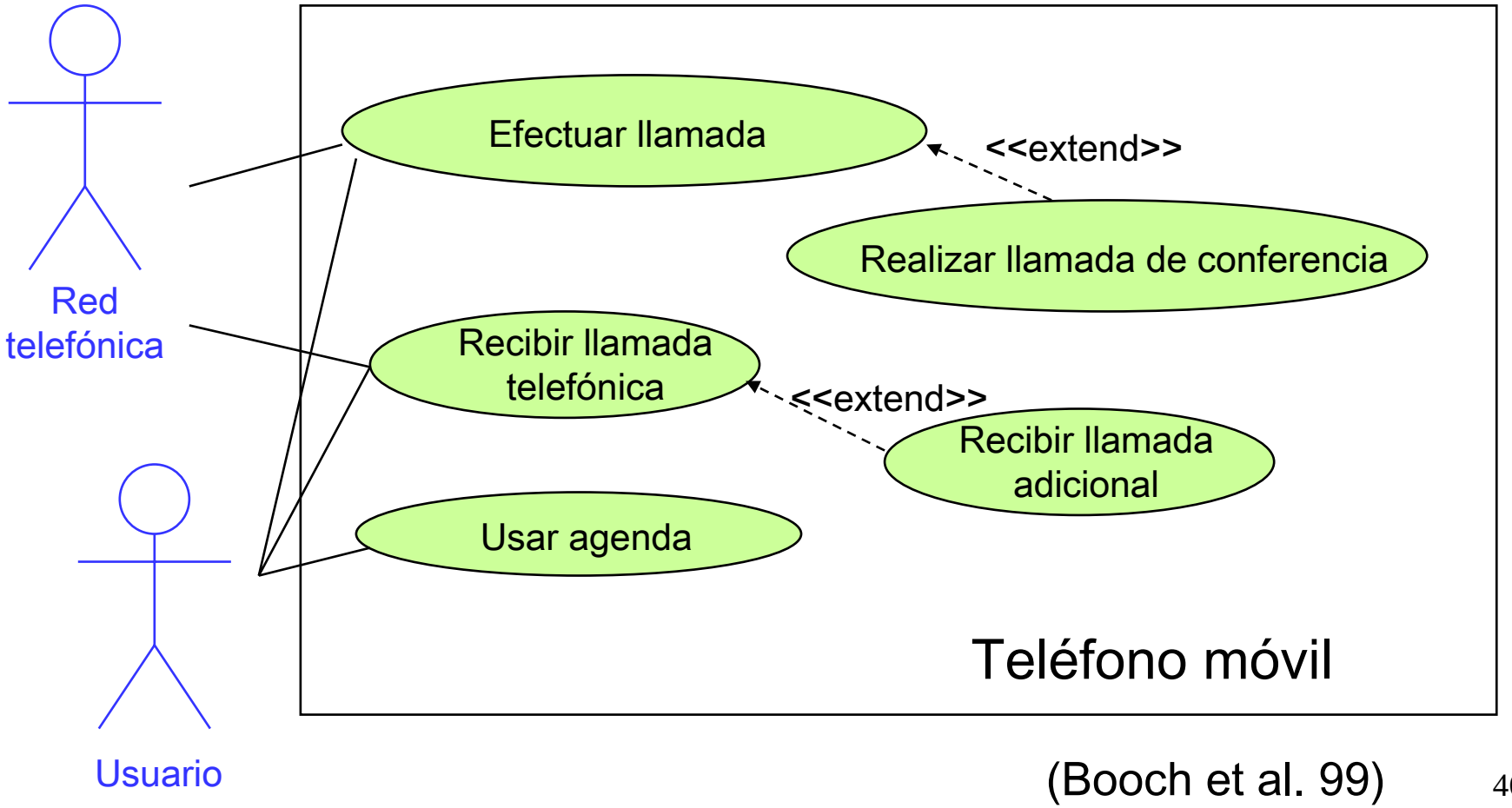
Éxito de los casos de uso

- Concebidos por I. Jacobson - Objectory/OOSE (Jacobson *et al.* 92)
- Presentes en casi cualquier nuevo método de desarrollo de software.
- Incluidos en UML y Métrica 3.
- Fuerte actividad investigadora en los últimos años.

Ejemplo Caso de Uso

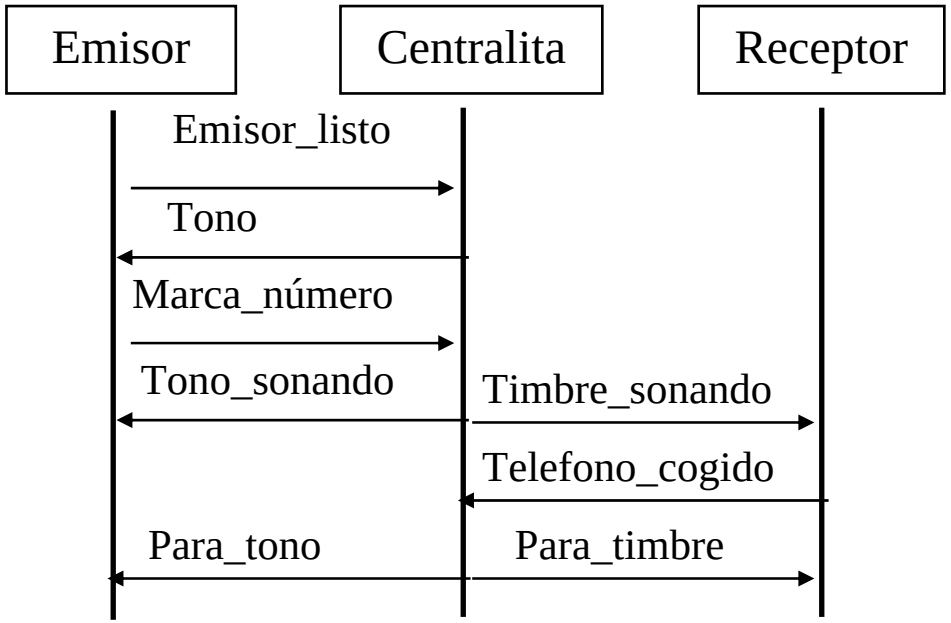


Ejemplo de Casos de uso

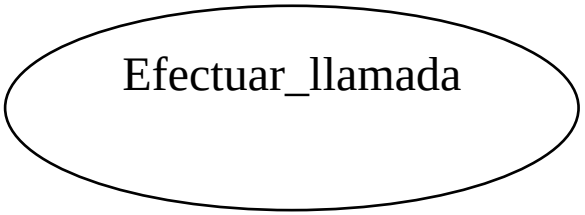


(Booch et al. 99)

Ejemplo de Casos de Uso



ESCENARIO



CASO DE USO

Otras definiciones de caso de uso

- *"Describe un conjunto de interacciones entre actores externos y el sistema en consideración orientadas a satisfacer un objetivo de un actor".* [D. Bredemeyer]

- *"Es una colección de posibles secuencias de interacciones entre el sistema en discusión y sus actores externos, relacionado con un objetivo particular".* [A. Cockburn]

- *"Es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor"*

  [UML]

Actores

Un actor representa un conjunto coherente de roles que juegan los usuarios de los casos de uso al interaccionar con el sistema.

- Roles jugados por personas, dispositivos, u otros sistemas.
- No forman parte del sistema
- Inician la ejecución de los casos de uso

Actores

- Un usuario puede jugar diferentes roles.
- En la realización de un caso de uso pueden intervenir diferentes actores.
- Un actor puede intervenir en varios casos de uso.
- Identificar casos de uso mediante actores y eventos externos.
- Un actor necesita el caso de uso y/o participa en él.

Actores

A. Cockburn distingue dos tipos de actores:

- **Primarios**:

  Requieren al sistema el cumplimiento de un objetivo

- **Secundarios**:

  El sistema necesita de ellos para satisfacer un objetivo

Propiedades de los casos de uso

- **Son iniciados por un actor** con un **objetivo** en mente y es completado con éxito cuando el sistema lo satisface

  ¿y si el caso de uso se inicia automáticamente?

  $\Rightarrow$ Actor “Sistema” o “Tiempo”
- Puede incluir secuencias **alternativas** que llevan al **éxito** y **fracaso** en la consecución del objetivo.
- El sistema es considerado como una **“caja negra”** y las interacciones se perciben desde fuera.
- El conjunto completo de casos de uso especifica todas las posibles formas de usar el sistema, esto es el **comportamiento requerido**.

Encontrar los casos de uso

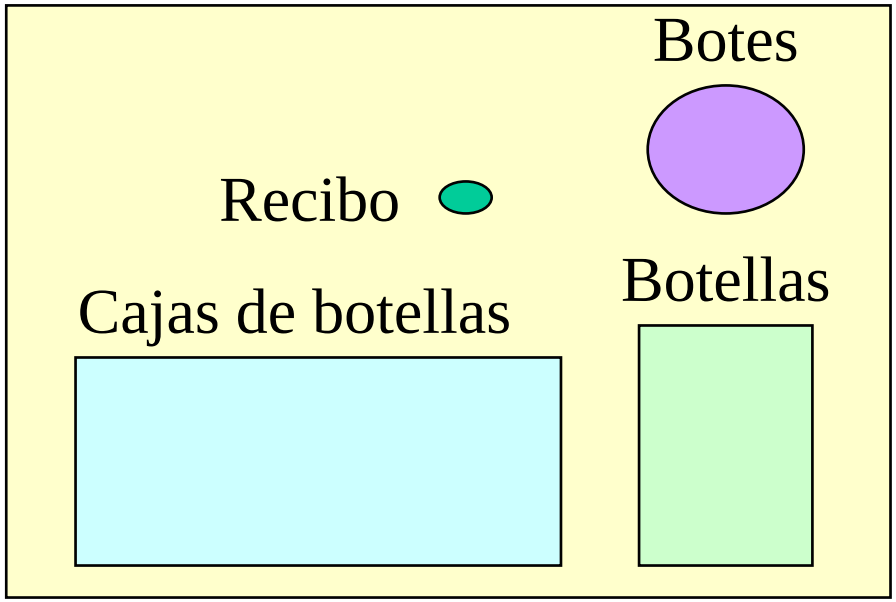
- Es útil encontrar primero los actores
- Se puede diferenciar entre (Fowler):
  - Objetivos del usuario:

    "formatear un documento"
  - Interacciones del sistema:

    "def. un estilo", "mover un estilo a otro doc."
- Guía útil: primero buscar los objetivos del usuario, y luego cubrir cada objetivo con interacciones del sistema.

Casos de uso. Ejemplo

- Máquina de reciclado:



Extraído de (Jacobson 92)

Casos de uso. Ejemplo

Máquina de reciclado de botes, botellas y envases de plástico

- Como cada ítem tiene precios y dimensiones distintas el sistema debe identificar el tipo de ítem que acaba de recibir
- El sistema registra el número de ítems recibidos y, si el cliente pide un recibo, imprime el número de ítems recibidos, su tipo, los precios parciales y el total que será devuelto al cliente en la caja, al presentar ese recibo impreso
- Un operador puede, al final del día, solicitar un listado de todos los ítems recuperados ese día
- El operador también puede cambiar la información del sistema (precios, tipos, etc.)

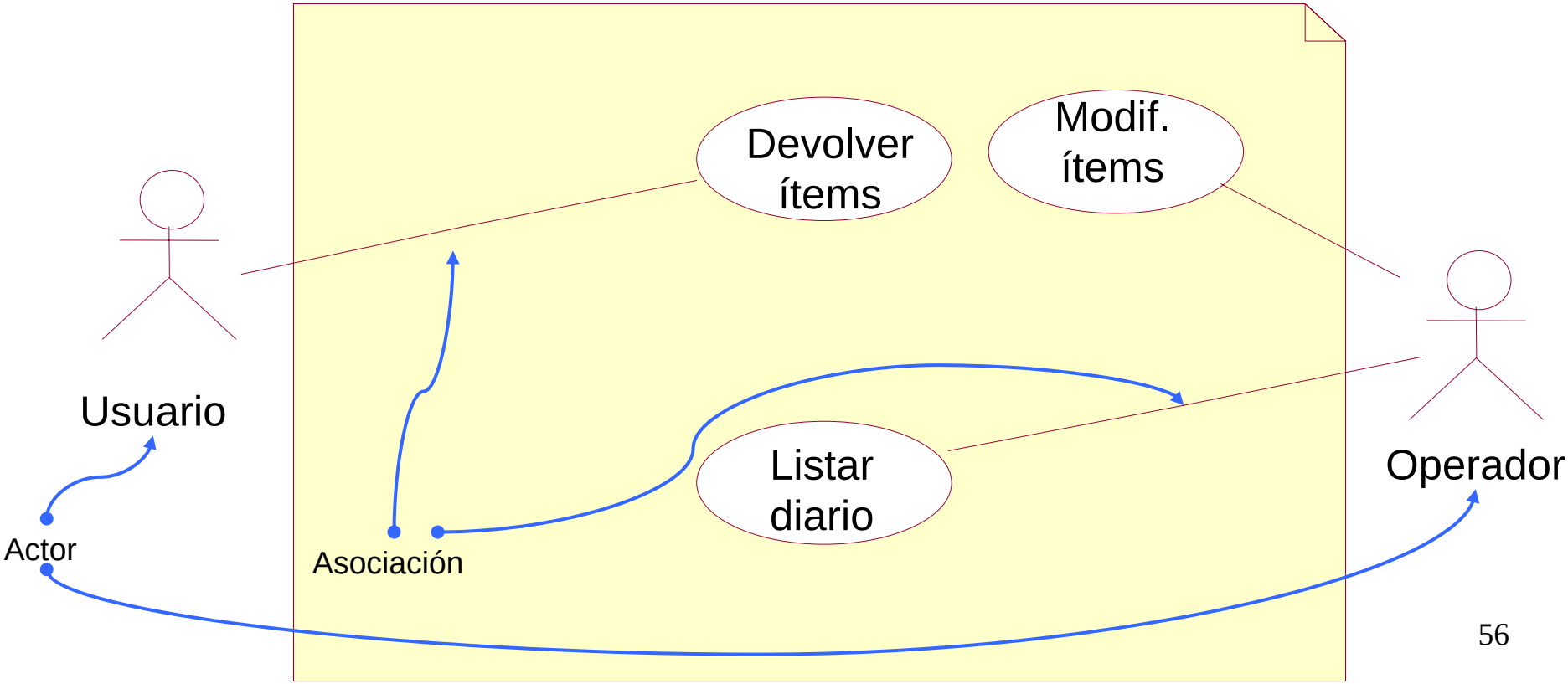
Casos de uso. Ejemplo

Primero determinar los actores: Usuario y Operador

Después determinar los objetivos de los actores:

Usuario: puede devolver ítems (el CdU incluye desde la devolución de ítems a la impresión del recibo)

Operador: puede pedir listado diario o bien modificar información de ítems



Escenarios y Casos de Uso

- Un caso de uso describe un conjunto de secuencias de interacciones o **escenarios:** flujo principal y flujos alternativos o excepcionales.
- Un escenario es una instancia de un caso de uso.
- Escenarios principales vs. Escenarios secundarios.
- Especificación con diagramas de secuencia.

Escenarios y Casos de Uso

Interactuar con ATM

CASO DE USO

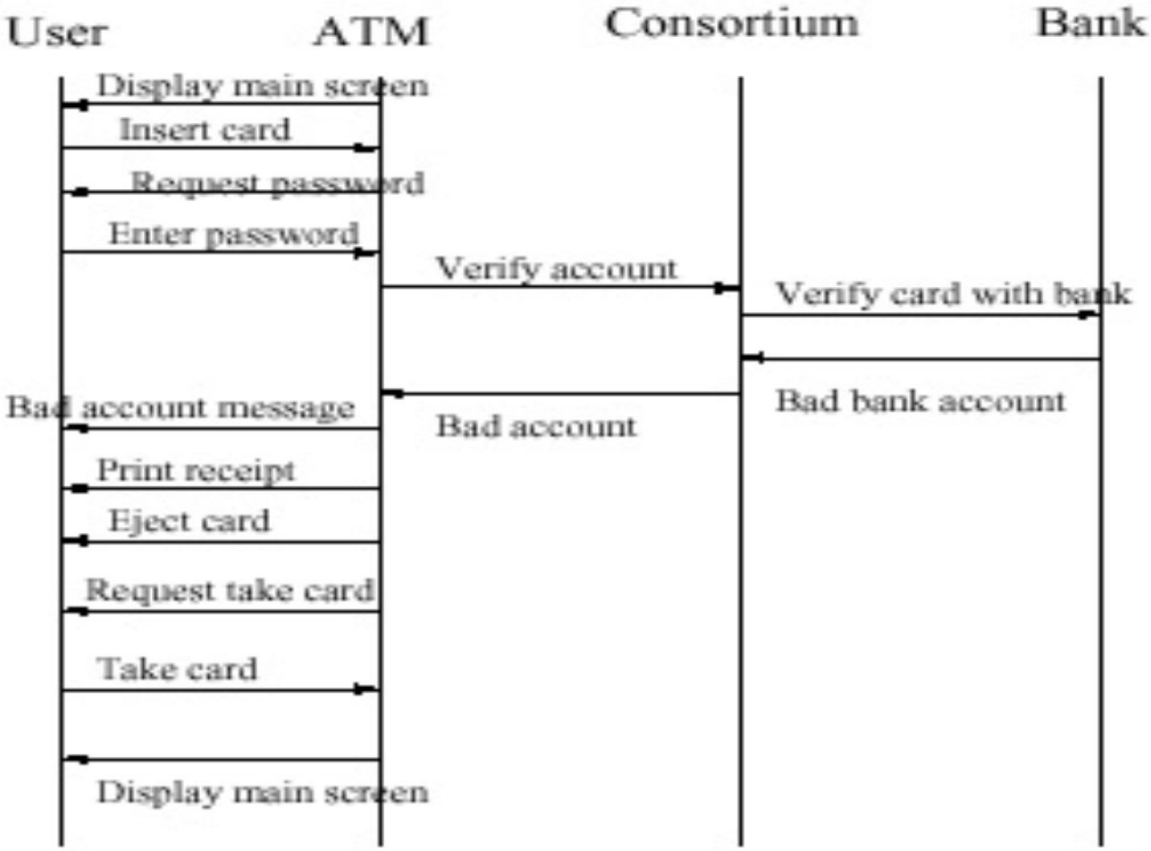


Figure 3: Interaction with an ATM (SD1).

ESCENARIO

Descripción de un caso de uso

- En cada momento de la especificación de cdu, una representación (Larman):
  - Descripción breve (párrafo en lenguaje natural)
  - Descripción informal (párrafo en lenguaje natural para escenario ppal. y alternativos)
  - Descripción completa (plantilla)
- Describir el flujo de eventos
  - Texto estructurado informal
  - Texto estructurado formal (pre y postcondiciones)
  - Notaciones gráficas: Diagramas de Secuencia
- Debe ser legible y comprensible para un usuario no experto.
- Debe indicarse: inicio y final, actores, objetos que fluyen, flujo principal y flujos excepcionales.

Descripción de un caso de uso

Comprar articulos (en un terminal de punto de venta)

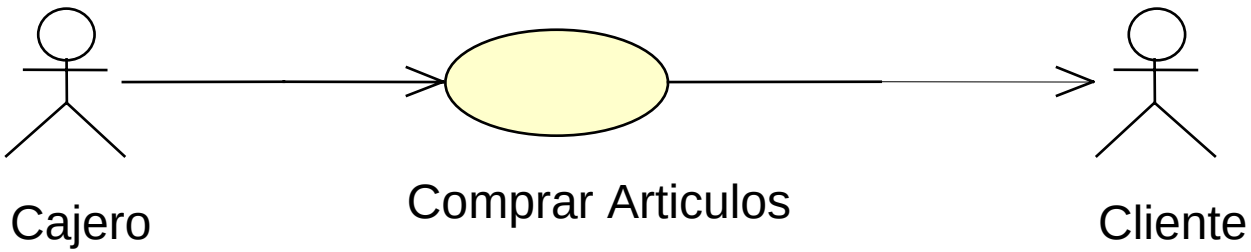
Flujo Principal: Un cliente llega al TPV con un conjunto de articulos. El Cajero registra los artículos y se genera un ticket. El cliente paga en efectivo y recoge los artículos.

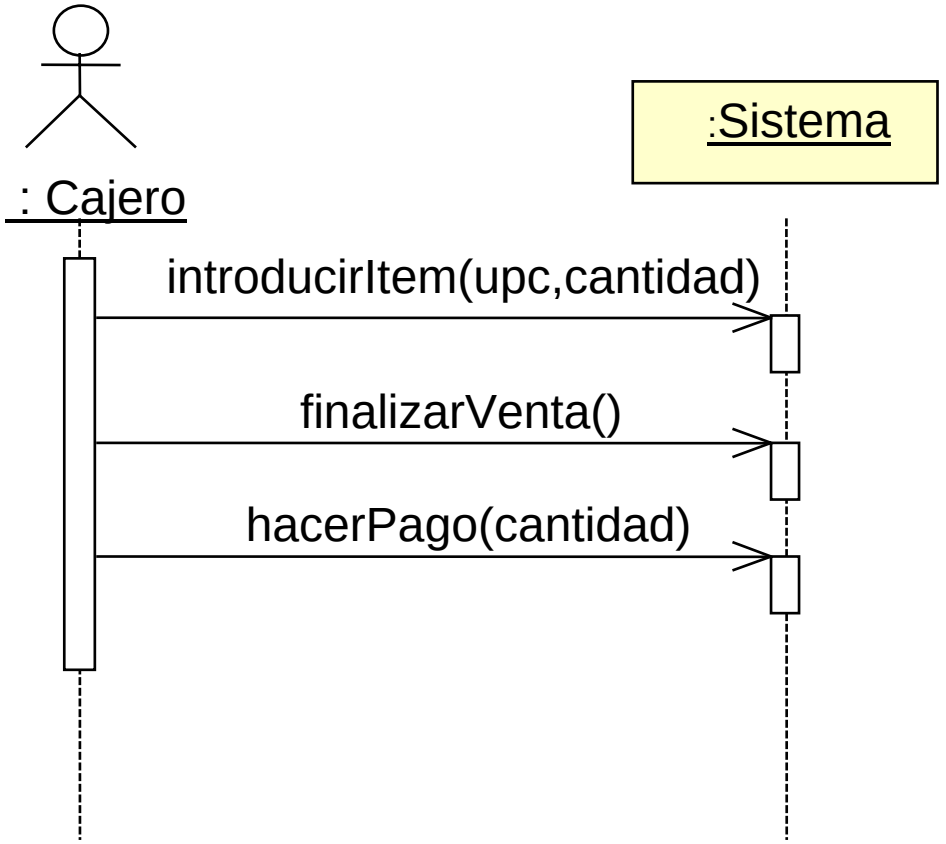
1. El cliente llega al TPV con los artículos.
2. El cajero registra el identificador de cada artículo.
3. El sistema obtiene el precio de cada artículo y añade la información a la transacción de venta.
4. Al acabar el cajero indica la finalización de la introducción de artículos.
5. El sistema calcula el total de la compra y lo muestra.

Descripción de un caso de uso

Comprar articulos (en un terminal de punto de venta)

6. El Cajero le dice al cliente el total.
7. El cliente realiza el pago.
8. El cajero registra la cantidad de dinero recibida.
9. El sistema muestra la cantidad a retornar al cliente y genera un recibo.
10. El cajero deposita el dinero recibido y saca la cantidad a devolver que entrega al cliente junto al ticket de compra.
11. El sistema almacena la compra completada.
12. El cliente recoge los artículos comprados.


Cajero
Comprar Articulos
Cliente


: Cajero
:Sistema
introducirItem(upc,cantidad)
finalizarVenta()
hacerPago(cantidad)

Descripción de un caso de uso

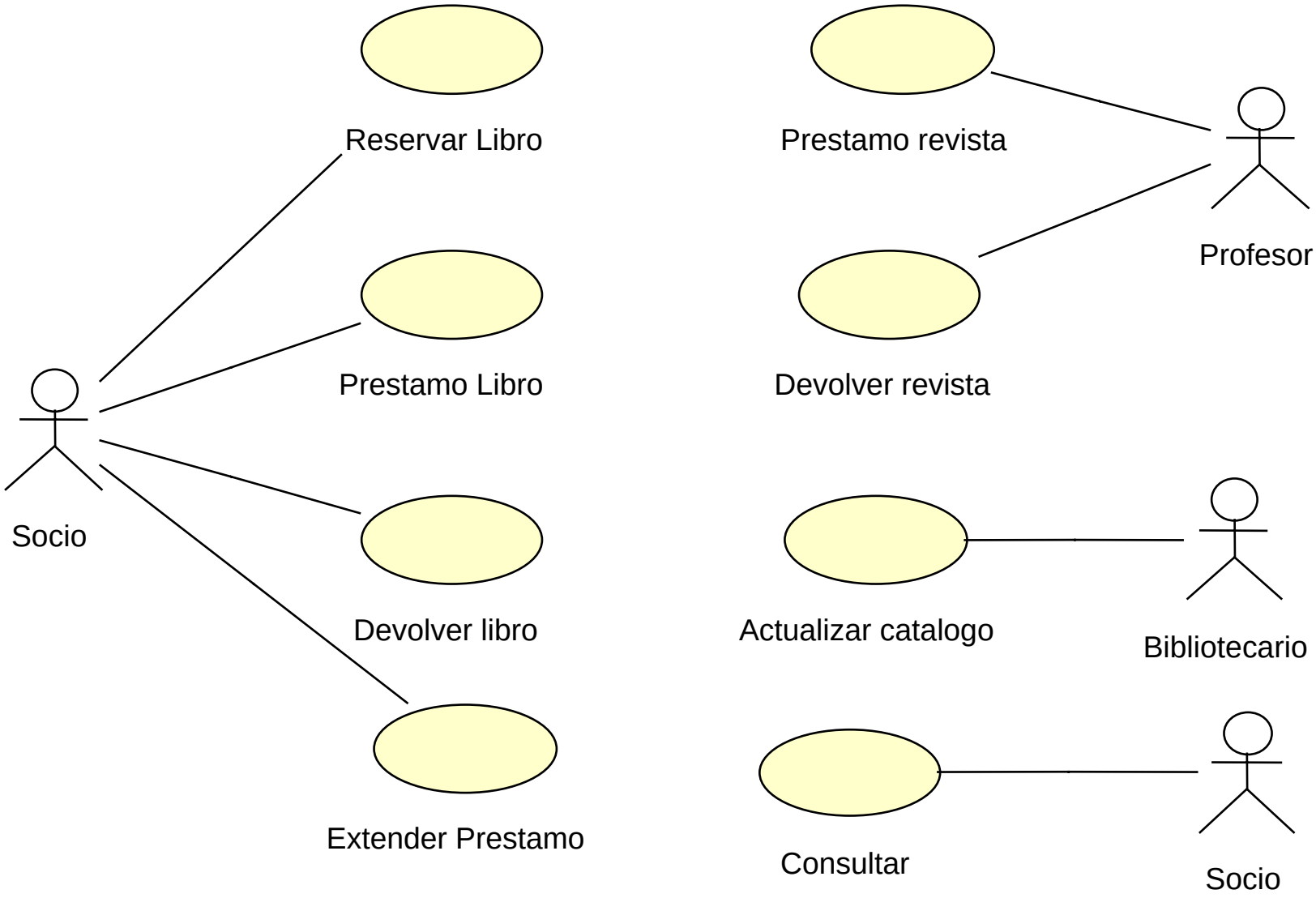
Validar Usuario (contexto de un cajero automático)

Flujo Principal: El sistema pide el NIP. El cliente lo introduce a través del teclado y pulsa Enter. El sistema comprueba la validez del NIP. Si es válido el sistema acepta la entrada y finaliza el caso de uso.

Flujo Excepcional: El cliente puede cancelar la transacción en cualquier momento, pulsando el botón Cancelar, reiniciando el caso de uso.

Flujo Excepcional: Si el NIP introducido es inválido entonces se reinicia el caso de uso. Si esto ocurre tres veces, el sistema cancela la transacción completa y se queda con la tarjeta.

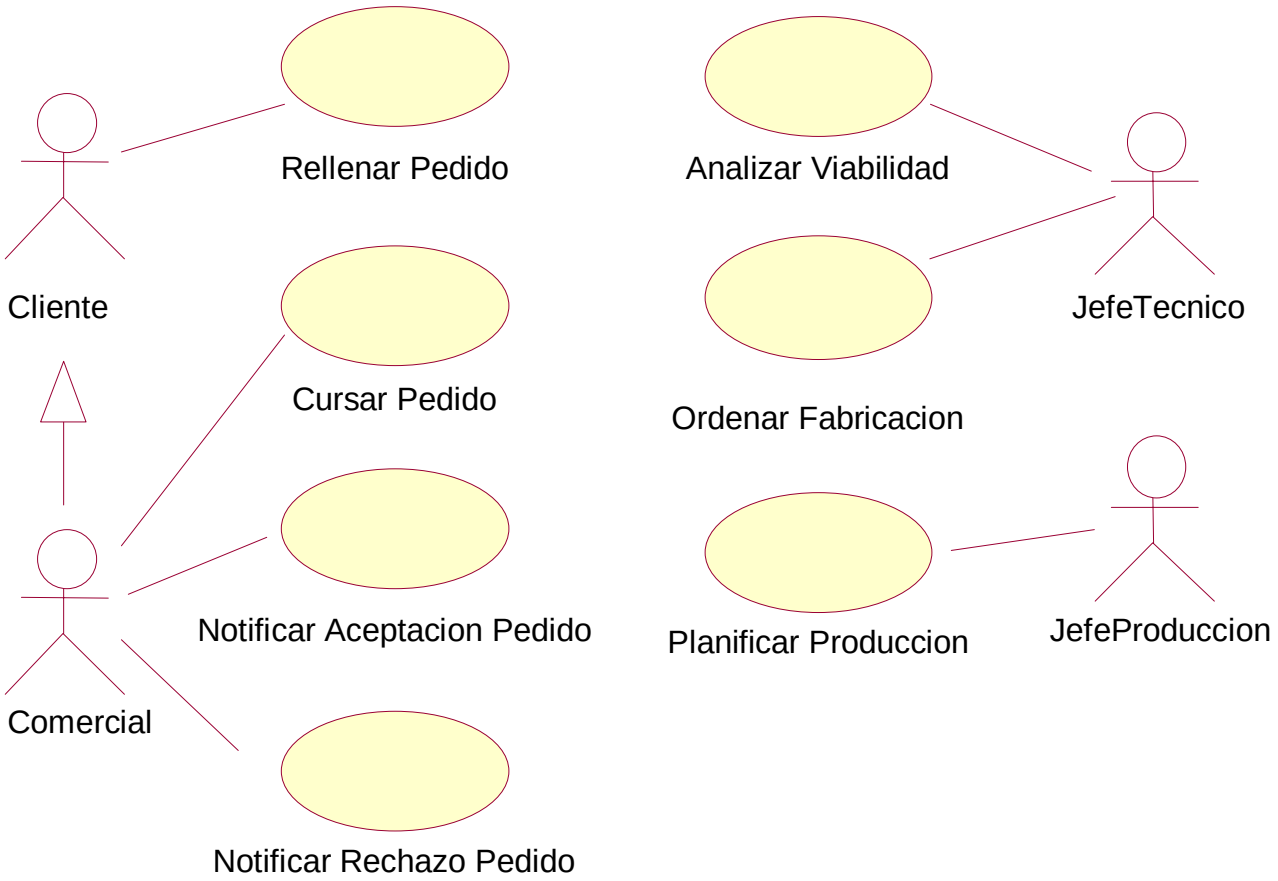
Diagrama de Casos de uso



Algunas consideraciones…

- Cdu “consulta”
- Cdu “CRUD”
  - Create/Read/Update/Delete
  - cdu “Manage X”
- ¿Son “universales” los cdu?
  - No (Larman 2003)
  - La “Especificación Suplementaria” puede completar la especificación de requisitos funcionales

Diagrama de Casos de Uso



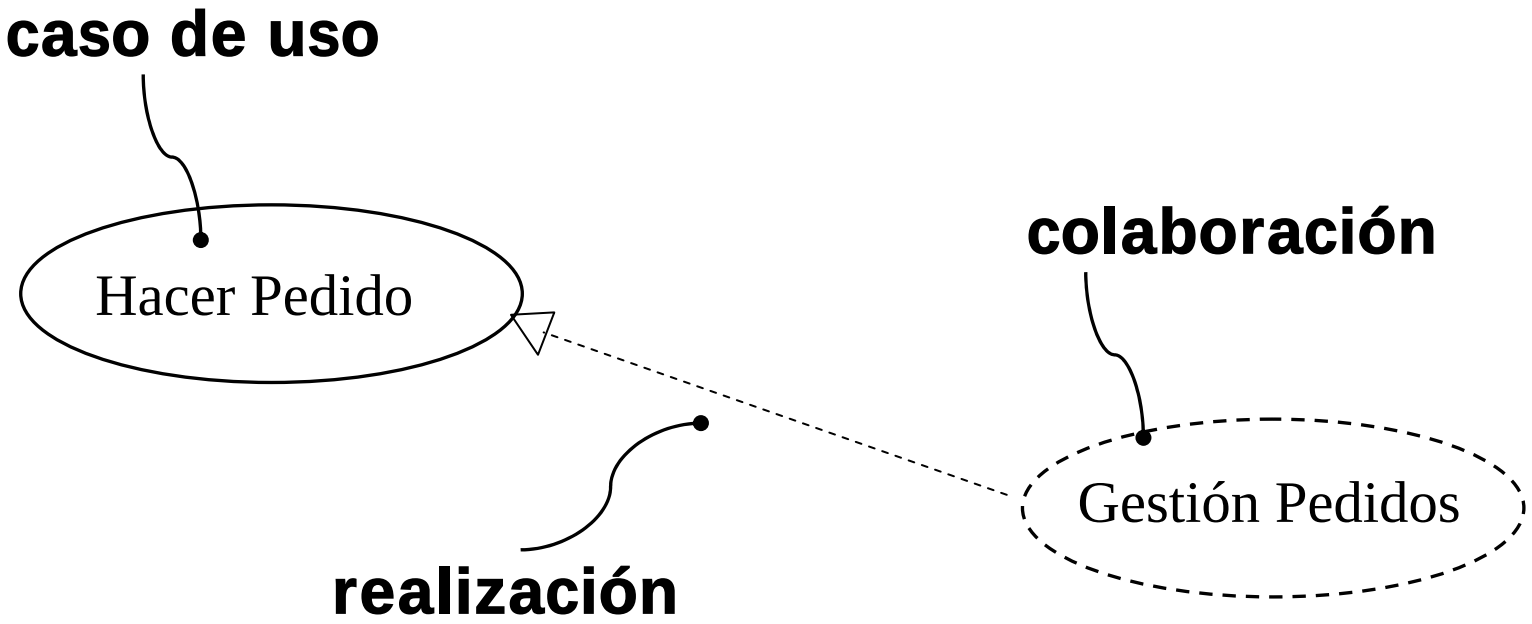
⇒ ¿Son cdu del mismo nivel que los de la biblioteca?

Casos de uso y Colaboraciones

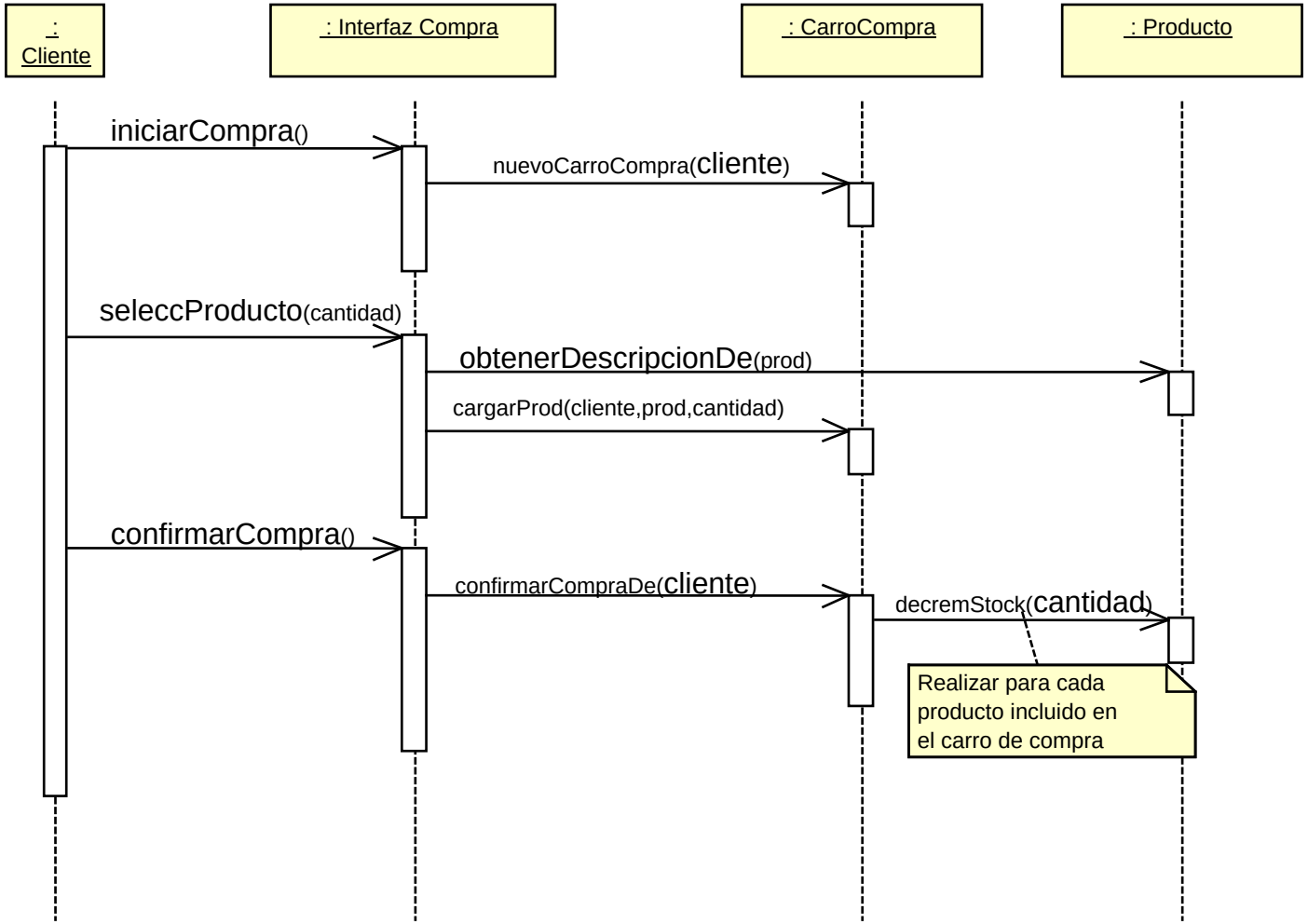
- Con un caso de uso se describe un comportamiento esperado del sistema, pero no se especifica cómo se implementa.
- Una caso de uso se implementa a través de una ***colaboración:***

  "Sociedad de clases y otros elementos que colaborarán para realizar el comportamiento expresado en un caso de uso"
- Una colaboración tiene una parte estática (diagramas de clases) y una parte dinámica (diagramas de secuencia o de colaboración/comunicación).

Casos de uso y Colaboraciones



Escenario del caso de uso “Realizar Compra”



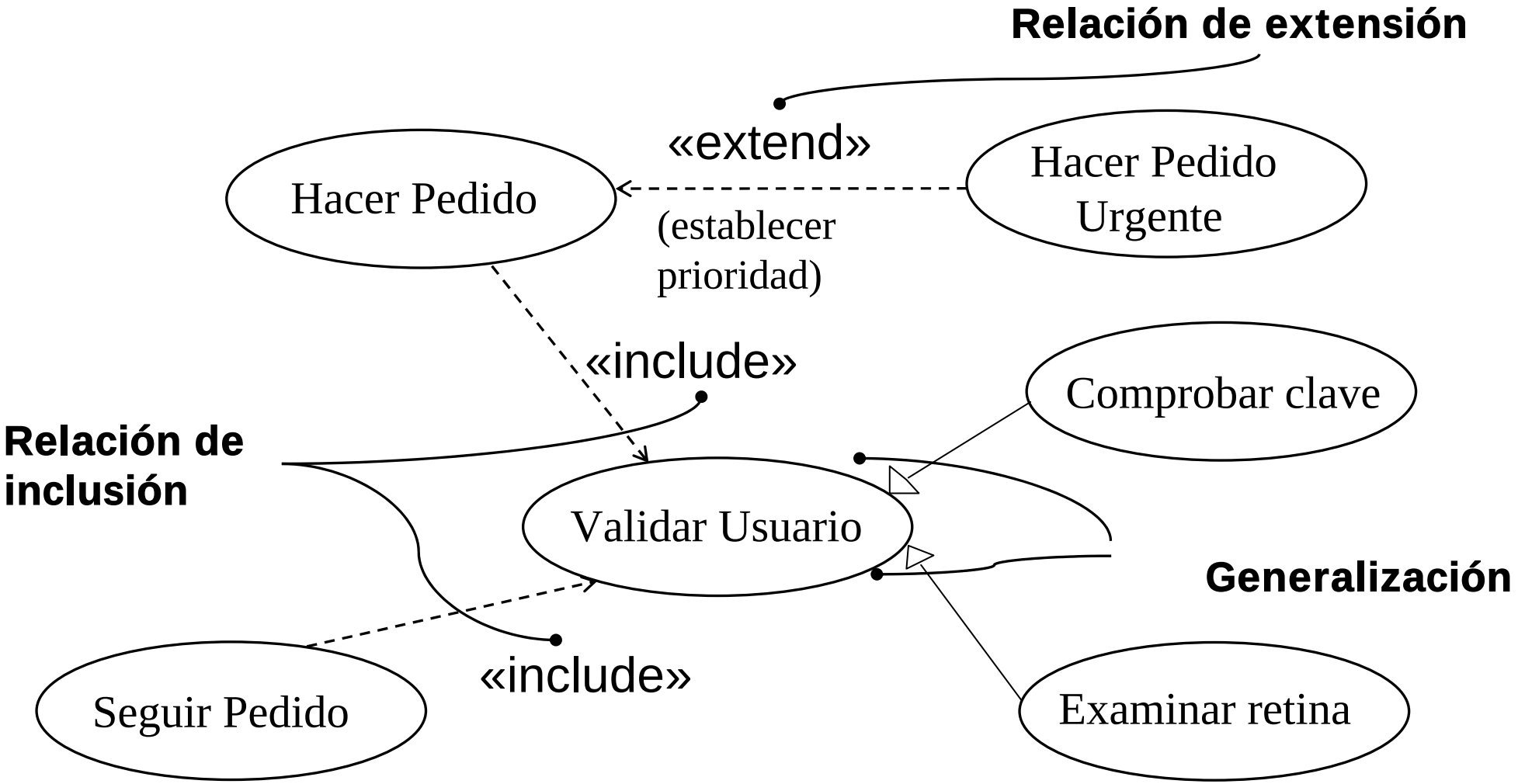
Casos de uso y Colaboraciones

"El objetivo de la arquitectura del sistema es encontrar el conjunto mínimo de colaboraciones bien estructuradas, que satisfacen el comportamiento especificado en todos los casos de uso del sistema"

Organización de Casos de uso

- Tres tipos de relaciones:
  - ***Generalización***
    - ***Un c.d.u. hereda el comportamiento y significado de otro***
  - ***Inclusión***
    - ***Un c.d.u. base incorpora explícitamente el comportamiento de otro en algún lugar de su secuencia.***
  - ***Extensión***
    - ***Un c.d.u. base incorpora implícitamente el comportamiento de otro c.d.u. en el lugar especificado indirectamente por este otro c.d.u.***

Ejemplo


Relación de extensión
«extend»
Hacer Pedido
Hacer Pedido Urgente
(establecer prioridad)
«include»
Relación de inclusión
Comprobar clave
Validar Usuario
Generalización
«include»
Seguir Pedido
Examinar retina

Relación de inclusión

- Permite factorizar un comportamiento en un caso de uso aparte y evitar repetir un mismo flujo en diferentes casos de uso.
- Ejemplo caso de uso *"Seguir Pedido":*

  "Obtener y verificar el número de pedido. **Include** (*Validar usuario*). Examinar el estado de cada parte del pedido y preparar un informe para el usuario".

Relación de extensión

- El caso de uso base incluye una serie de puntos de extensión.
- Sirve para modelar
  - la parte opcional del sistema
  - un subflujo que sólo se ejecuta bajo ciertas condiciones
  - varios flujos que se pueden insertar en un punto
- Ejemplo caso de uso "*Hacer Pedido*":

  " **Include** (*Validar usuario*). Recoger los items del pedido del usuario. (*establecer prioridad*). Enviar pedido para ser procesado.

Obtención de casos de uso

1) Identificar los usuarios del sistema.
2) Encontrar todos los roles que juegan los usuarios y que son relevantes al sistema.
3) Para cada role identificar todas las formas (objetivos) de interactuar con el sistema.
4) Crea un caso de uso por cada objetivo.
5) Estructurar los casos de uso.
6) Revisar y validar con el usuario.

Plantilla para casos de uso (D. Coleman)

Caso de Uso	identificador / historia
Descripción	objetivo a conseguir
Actores	lista de actores
Asunciones	Condiciones que deben cumplirse
Pasos	interacciones entre los actores y el sistema necesarias para obtener el objetivo
Variaciones	(opcional) cualquier variación en los pasos
No-funcional	(opcional) lista requisitos no funcionales
Cuestiones	Lista de cuestiones que permanecen por resolver

Plantilla para casos de uso (D. Coleman)

Ejemplo campo "*pasos*":

1. Operador es notificado de problema en la red
2. Operador inicia una sesión de reparación
3. REPETIR
    3.1. Operador ejecuta aplicación diagnósticos en la red.
    3.2. Operador identifica elementos que deben cambiarse y sus nuevos valores para los parámetros.
    3.3. EN PARALELO
        3.3.1. Ingeniero mantenimiento comprueba elementos en la red ||
        3.3.2. Ingeniero mantenimiento envía informe fallo
    4. Operador cierra sesión

Plantilla para casos de uso (D. Coleman)

Relación de **uso**: PERFORM c.d.u.

Relación de **extensión**:

Caso de uso extensión id. c.d.u. extends id. c.d.u.

Cambio	objetivo que debe conseguirse
Pasos	...
Variaciones	...
No funcional	...
Cuestiones	...

Ejemplo

Caso de Uso	Ordenar Fabricación
Descripción	Se crearán órdenes de trabajo para cada producto solicitado en el pedido, y serán enviadas al jefe de producción para su planificación.
Actores	Jefe técnico
Asunciones	- Es viable la fabricación de cada producto solicitado en el pedido. - Existe una plantilla de fabricación para cada producto solicitado.
Pasos	1. REPETIR 1.1 Obtener un producto del pedido. 1.2 Buscar la plantilla de fabricación asociada al producto. 1.3 Crear la orden de trabajo. 1.4 Almacenar la orden de trabajo con el estado *pendiente*.
Variaciones	-
Req. No Funcionales	-
Cuestiones	-

Plantilla Caso de Uso (A. Cockburn)

Sistema	Compañía Seguros
Actor principal	Asegurado
Objetivo	Cobrar seguro accidente

1. Asegurado envía reclamación
2. Compañía verifica que asegurado tiene una póliza válida
3. Compañía asigna agente
4. Agente verifica todos los detalles son conformes el contrato
5. La compañía paga al asegurado

Plantilla Caso de Uso (A. Cockburn)

Extensiones

1a. Datos del parte incompletos
    1a1. Compañía requiere datos
    1a2. Asegurado envía datos

2a. Asegurado no tiene una póliza válida
    2a1. Compañía rechaza reclamación, avisa al asegurado.

3a. No hay agentes disponibles
    3a1. ¿qué hace la compañia?

….

Plantilla Caso de Uso (A. Cockburn)

Variaciones

1. Asegurado
    a) una persona
    b) otra compañía de seguros
    c) el gobierno

2. Pago
    a) transferencia
    b) dinero en efectivo
    c) cheque

¿Por qué casos de uso?

- “Ofrecen un medio sistemático e intuitivo para capturar los requisitos funcionales, centrándose en el valor añadido para el usuario”
- Dirigen todo el proceso de desarrollo puesto que la mayoría de actividades (planificación, análisis, diseño, validación, test,..) se realizan a partir de los casos de uso.
- Mecanismo importante para soportar “trazabilidad” entre modelos.

Crítica a los casos de uso (B.Meyer, E. Berard)

- Los casos de uso no son adecuados en el modelado orientado a objetos porque:

i) Favorecen un enfoque funcional, basado en procesos

ii) Se centran en secuencias de acciones

iii) Se basan en los escenarios actuales del sistema

“Solo deben ser usados por equipos expertos en desarrollo OO”

- Utiles para validación

Utilidad de los casos de uso

- Hay **consenso** en considerar casos de uso como esenciales para capturar requisitos y guiar el modelado.
- Pero existe mucha **confusión** sobre cómo usarlos.
- Diferentes opiniones sobre el número de casos de uso conveniente:
  - 20 para un proyecto 10 personas/año (Jacobson)
  - 100 para un proyecto 10 personas/año (Fowler)
  - depende de la granularidad

Granularidad (M. Fowler)

- Diferente granularidad
- Un caso de uso puede describir:
  - Un objetivo o propósito del usuario
  - Una interacción con el sistema
- Un objetivo implica una o más interacciones.
- Primero construir un caso de uso por cada objetivo del usuario.
- Después escribir casos de uso para cada interacción que corresponda a un objetivo.

Granularidad (A. Cockburn)

- Organización
  - Objetivo estratégico para la empresa, de mucho valor
- Sistema (objetivo del usuario)
  - Funcionalidad del sistema, tarea del usuario o "proceso de negocio elemental"
- Subfunciones
  - Un paso en la descripción de un caso de uso (validar, buscar, log on)

Tipos de casos de uso (Larman)

- Descripción breve, informal o completa
  - Descripción muy general o “conversación” entre actor y sistema.
- Primarios, Secundarios u Opcionales
  - Según su importancia en el sistema
- Esenciales vs. Reales
  - Según su grado de abstracción
  - Un caso de uso real describe el proceso en términos de su diseño actual, teniendo en cuenta tecnología de E/S.

Recomendaciones (E. Berard)

- Un caso de uso no debe considerar cuestiones de implementación.
- Conveniencia de una herramienta para la gestión de los casos de uso.
- Encontrar contradicciones entre casos de uso.
- Preocupación por mantener la validez y consistencia del conjunto de casos de uso.
- ¿Cómo se comprueba que los casos de uso incluyen toda la funcionalidad del sistema?

Recomendaciones (E. Berard)

- Cada compañía debe tener un manual sobre uso de los casos de uso.
- ¿A qué nivel de detalle se describe un caso de uso?
- ¿Qué granularidad es apropiada para un caso de uso?
  - “Pinchar botón”, “Añadir Empleado”,.. ¿son casos de uso?

Recomendaciones (M. Fowler)

- Cuidado con el empleo de la relación “uses”

  ¡NO HAGAS UNA DESCOMPOSICION FUNCIONAL!
- No abusar de la estructuración de casos de uso
- No es necesario aplicar la abstracción
  - ¡USA CASOS DE USO CONCRETOS!

Recomendaciones A. Pols

- Primero establecer los objetivos del proyecto.
- Después identificar actores y sus responsabilidades, y las tareas que ejecutan son los casos de uso.
- Contrastar casos de uso frente a los objetivos.
- Cuidado con la ambigüedad
- No profundizar en la descripción de un caso de uso
- No te preocupes en exceso de la notación

Recomendaciones (A. Pols)

- Evita redes complicadas de casos de uso: Cuidado con las relaciones *include* y *extend*.
- No considerar la interfaz del usuario
- Los casos de uso sólo consideran los requisitos funcionales del proyecto, hay que añadir los no funcionales.

Especificación de requisitos

- La ERS (*Especificación de Requisitos del Software*) puede estar formada por:
  - Diagrama de casos de uso
  - Modelo del dominio (modelo conceptual)
  - (Para cada caso de uso)

    Descripción textual (usando una plantilla)

    Descripciones de las interfaces de usuario
  - Requisitos no funcionales

Casos de Uso y Componentes

- Pensar en componentes lo antes posible
  - Ahorrar esfuerzo
  - Negociar los requisitos
- Si se producen componentes
  - Asociarle su propio caso de uso