

ANALISIS Y DISEÑO DE SISTEMAS (3303)

Notaciones Gráficas para el Modelado de Sistemas

Unified Modeling Language

Mg. Marcela Daniele

Departamento de Computación

Facultad de Ciencias Exactas, Físico-Químicas y Naturales

Universidad Nacional de Río Cuarto





EL ARTE DE MODELAR

- ✓ ¿Qué es un Modelo?
- ✓ ¿Porqué Modelar?
- ✓ Ventajas de los Modelos
- ✓ Notaciones Gráficas:
 - DFD (Diagrama de Flujo de Datos)
 - UML (Unified Modeling Language)
 - Diagrama de Clases
 - Diagrama de Objetos
 - Diagrama de Actividades
 - Diagrama de Estados
 - Diagrama de Casos de Uso
 - Diagrama de Secuencia



EL ARTE DE MODELAR

¿ Qué es un modelo?

**Un modelo es una interpretación
simplificada de la realidad**

*Los modelos describen abstracciones que incluyen lo esencial
de un problema complejo y facilitan su comprensión.*

Un sistema puede ser descrito con modelos desde
diferentes perspectivas:

- Estructural, destaca la organización del sistema.
- Comportamiento, resalta la dinámica del sistema.

*A mayor complejidad del problema mayor será la
necesidad de utilizar buenas técnicas de modelado.*



¿ Porqué modelar?

Los modelos de software nos permitirán:

- ✓ Administrar la complejidad
- ✓ Detectar errores u omisiones tempranamente en el ciclo de vida
- ✓ Mejorar la comunicación con todos los participantes (stakeholders)
- ✓ Entender los requerimientos
- ✓ Conducir la implementación
- ✓ Entender el impacto de los cambios
- ✓ Asignar recursos con eficiencia



EL ARTE DE MODELAR

Ventajas de los Modelos

- **Visualizar un sistema** desde varias perspectivas y facilitar su desarrollo.
- **Especificar la estructura y el comportamiento** del sistema y proporcionar plantillas que guíen su construcción.
- **Generar un medio de comunicación** entre desarrolladores y usuarios.
- **Documentar el desarrollo del sistema**, desde el planteo mismo del problema hasta su implantación y mantenimiento.

UML (Unified Modeling Language)

¿ QUE ES UML ?

Es un LENGUAJE de modelado estándar que permite **visualizar, especificar, construir y documentar** los **artefactos** de un sistema de software.

Artefacto: representa información producida o utilizada por PDS



Para representar un MODELO es importante elegir un lenguaje adecuado.

UML es adecuado para modelar cualquier sistema representado con objetos.



UML - QUE ES?

- Lenguaje *gráfico* para modelado de sistemas.
- Estándar abierto (OMG) Object Management Group.
(OMG: Produce y mantiene estándares para la industria de software)
- Soporta todas las etapas del ciclo de vida de desarrollo de software. (captura, análisis, diseño, implementación)
- Soporta distintas áreas de aplicación
(Sistemas distribuidos, tiempo real, Sistemas de información corporativos, Transporte, Distribución, Banca/Finanzas, Defensa/Espacio, Electromedicina, Ciencia, Telecomunicaciones, etc.)
- Soportado por herramientas

Astah, ArgoUML, Dia, Rational Rose, Together, ...



UML - VENTAJAS

- **Facilita la comunicación.**

Es un lenguaje: sintaxis + semántica.

- Reduce ambigüedades, a partir de la especificación de **modelos más precisos y completos.**
- Permite **Ingeniería Directa**. Desde un modelo UML se obtiene un código. (Java, C++,...)
- Permite **Ingeniería Inversa**. Desde una aplicación es posible obtener modelos UML.
- Permite **generar y actualizar la documentación** del sistema en cada etapa del ciclo de vida.



UML – Breve Historia

- Booch: método más expresivo durante las fases de construcción y diseño.
 - Jacobson: OOSE (Object Oriented Software Engineering). Buen soporte para la definición de Casos de Uso en la Captura de Requerimientos.
 - Rumbaugh: OMT (Object Modeling Technique). Más utilizado en el análisis.
- 1994. Booch y Rumbaugh (Rational). A fines del 95 aparece la 1ra. versión UML 0.8.
- 1995 + Jacobson. 1996 crean la versión UML 0.9.
- UML 1.0. Digital Equipment Corporation, HP, Oracle, IBM, Texas, Unisys, I/Logix, Intellicorp, ICOM, MCI, Microsoft, Rational.
- 1997 estandarizado por OMG (Object Management Group). UML no es propiedad de una empresa en particular y no existen los derechos de autor. UML 1.4.
- 2003. UML 2.0. Incorpora MDD (Model Driven Development) y MDA (Model Driven Architecture).



UML - Bloques de Construcción

➤ Elementos

Representan las partes de los modelos UML.

➤ Relaciones

Relaciones semánticas o estructurales entre elementos del modelo.

➤ Diagramas

Representación gráfica de un conjunto de elementos. Es un grafo conexo de nodos (**elementos**) y arcos (**relaciones**).

Un diagrama representa una vista de un sistema.



UML - Bloques de Construcción Elementos

- **Estructurales:** clase, interface, colaboración, caso de uso, clase activa, componente y nodo.
- **Comportamiento:** interacción, máquina de estados.
- **Agrupación:** paquetes.
- **Anotación:** notas explicativas.

UML - Bloques de Construcción Relaciones

- La clases colaboran unas con otras.
- Para modelar un sistema, es necesario identificar los elementos u objetos que conforman su vocabulario y la forma en que se relacionan unos con otros.
- Una relación es un tipo de conexión entre elementos. Pueden existir relaciones estructurales entre instancias, relaciones de uso, relaciones que conectan instancias de clases más generales con otras más especializadas.

UML - Bloques de Construcción Diagramas

Diagramas Estructurales

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Artefactos
- Diagrama de Despliegue

Diagramas de Comportamiento

- Diagrama de Caso de Uso
- Diagramas de Interacción:
 - Diagrama de Comunicación
 - Diagrama de Secuencia
- Diagrama de Estados
- Diagrama de Actividad

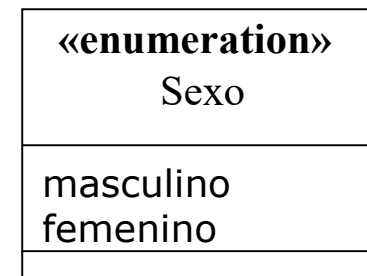
UML - Bloques de Construcción Diagramas

Vista	Lógica	Física
Estática	Clases	Artefactos
	Objetos	Despliegue
Dinámica	Casos de Uso Estados Actividad Interacción: Secuencia Comunicación	

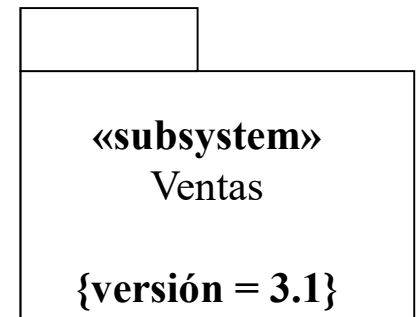
UML – Mecanismos de Extensibilidad

Permiten extender el lenguaje de manera controlada.

➤ **Estereotipos:** extienden el vocabulario, permiten crear nuevos tipos de bloques de construcción. Ej. «**subsystem**», «**exception**», «**enumeration**».



➤ **Valores Etiquetados:** extienden las propiedades de un bloque de construcción, permitiendo añadir nueva información en la especificación del mismo. Se denota con una cadena de caracteres entre llaves. Ej: {versión 3.1}.



➤ **Restricciones:** extienden la semántica de un bloque de construcción permitiendo añadir nuevas reglas o modificando las existentes. Se denota con una cadena de caracteres entre llaves o en una nota. Ej: {edad>20}

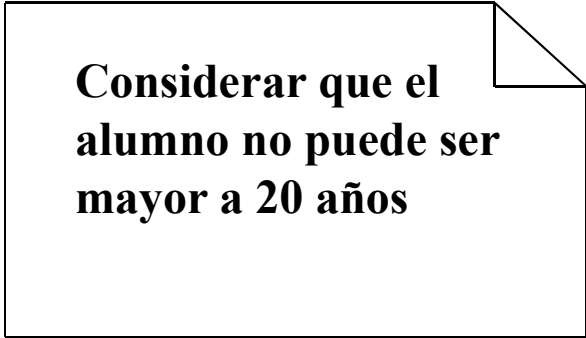
UML - Otros conceptos importantes

NOTA (adorno)

Es un símbolo gráfico para mostrar comentarios asociados a un elemento o a un conjunto de ellos.

Puede modelar información de requisitos, observaciones, revisiones y explicaciones.

Gráficamente, ejemplo:

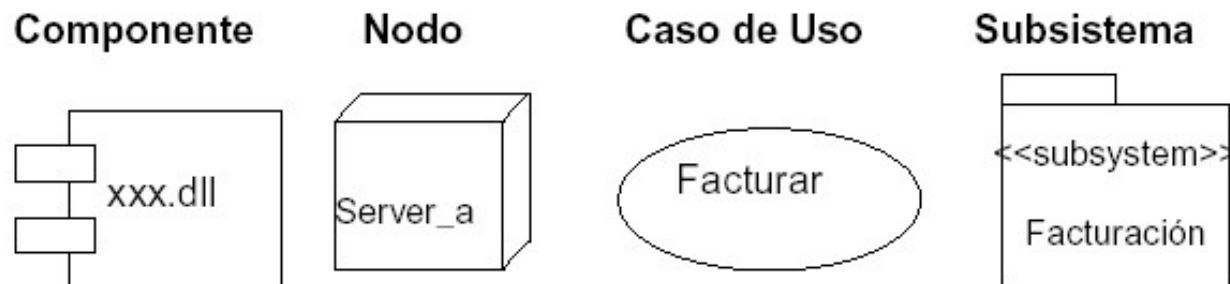
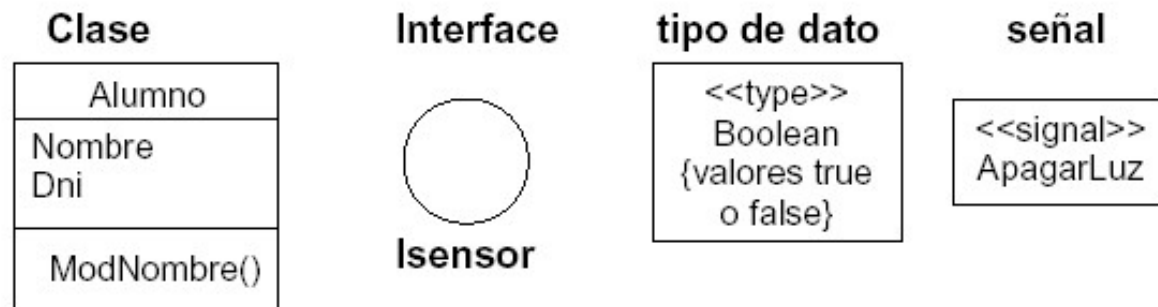
A UML Note symbol, which is a rectangular box with a folded top-right corner. It contains the text:

**Considerar que el
alumno no puede ser
mayor a 20 años**

UML - Otros conceptos importantes

CLASIFICADOR

Es un mecanismo de UML que sirve para describir características estructurales y de comportamiento de los modelos.



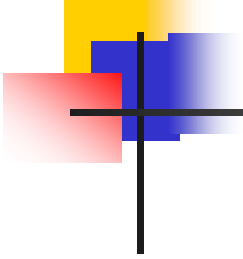


Diagrama de Clases

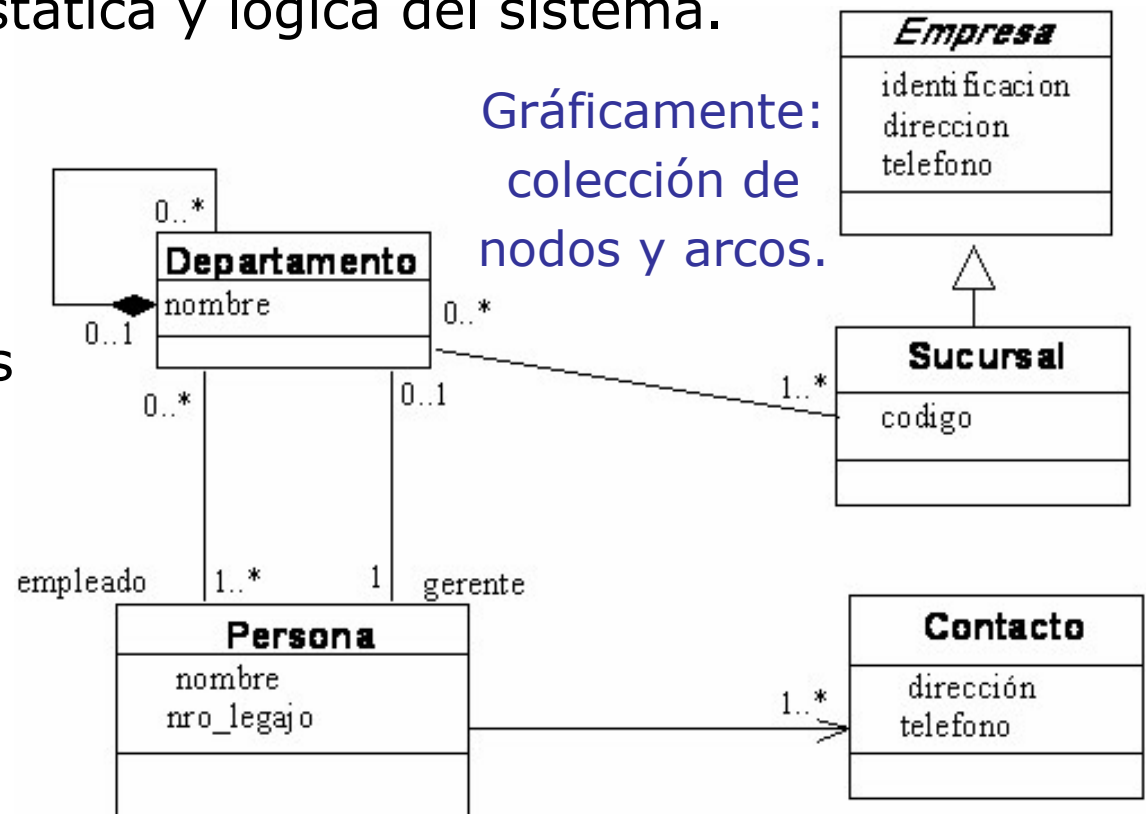
UML - Diagrama de Clases

➤ Muestra un conjunto de interfaces, clases, colaboraciones y relaciones y permite Visualizar, Especificar, Construir y Documentar modelos estructurales.

➤ Modela una vista estática y lógica del sistema.

➤ Colaboran para la construcción de sistemas ejecutables a través de ingeniería directa e inversa.

Gráficamente:
colección de
nodos y arcos.



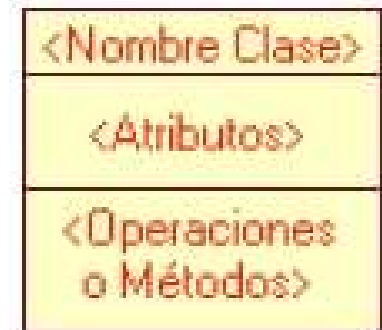
UML – Diagrama de Clases.

Elementos de un Diagrama de Clases

- Clases
- Relaciones
- Interfaces

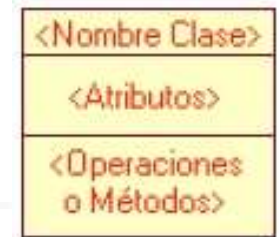
CLASE

- Bloque básico de construcción en los sistemas orientados a objetos.
- Una clase es una abstracción de un conjunto de objetos, que comparten atributos, operaciones y semántica.
- Gráficamente, una clase es representada por un rectángulo con tres compartimentos.



UML - Diagrama de Clases

CLASE



Nombre de la clase

- Cada clase debe tener un nombre que la distinga de las otras.
- Debe ser significativo respecto al concepto que la clase representa.

Atributos

- Un atributo es una propiedad con nombre y describe un rango de valores que las instancias de la propiedad pueden asumir.
- En un momento en el tiempo, un objeto de la clase tendrá valores específicos para cada uno de sus atributos.

Métodos u operaciones

- Representan la forma en que interactúa el objeto con su entorno.
- Una operación es un servicio que puede ser requerido a cualquier instancia de la clase para realizar un comportamiento.



Una clase puede tener cualquier # de atributos y operaciones.



UML - Diagrama de Clases

CLASE - VISIBILIDAD en Atributos y Métodos

- **public (+)**: todos los objetos enlazados pueden acceder al atributo o método.
- **private (-)**: sólo es accesible desde objetos que pertenecen a la misma clase.
- **protected (#)**: podrá ser accedido por métodos de la clase y de las subclases que se deriven (ver herencia). No será accesible desde fuera de la clase.

UML - Diagrama de Clases

CLASE - Responsabilidades

Son contratos u obligaciones de una clase

➤ En las primeras vistas de la clase, alcanza con definir sus responsabilidades en lenguaje natural.

➤ En vistas más detalladas, se necesita definir su estructura y comportamiento, para llevar a cabo esas responsabilidades.

➤ Cuando evoluciona el diseño y se comienza con la implementación, es necesario definir la visibilidad de los atributos y operaciones la clase realiza.

Alumno
Nombre Dni
Dar el nombre completo

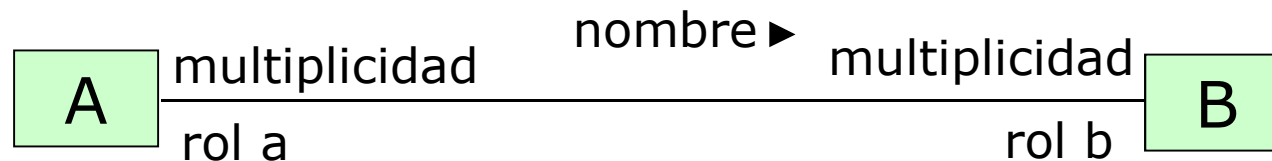
Alumno
+ Nombre: String - Dni:String
+ GetFullName()

UML - Relación de **ASOCIACION**

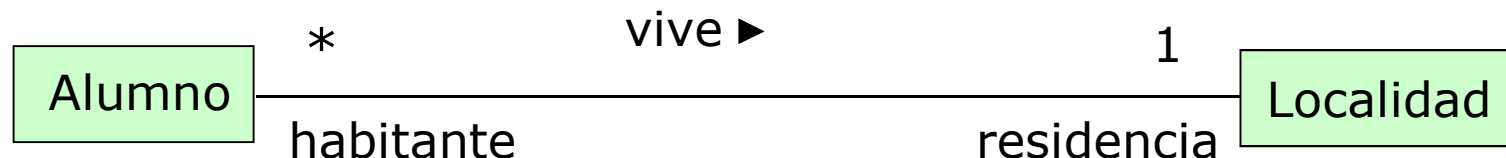
Relación estructural que describe conexiones entre objetos.

- Se denota por una línea sólida que conecta dos clases.

Puede incluir etiqueta o nombre, multiplicidad, rol y navegación.

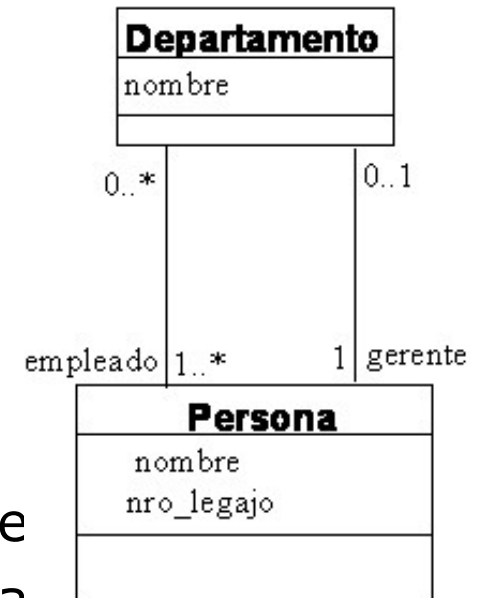


- El **nombre** describe la naturaleza de la relación.
 - ▶ indica la dirección del nombre.
- El **rol** define el papel que una clase juega sobre la otra.
- La **multiplicidad** determina cuantos objetos pueden estar conectados con otros en una instancia de la asociación.

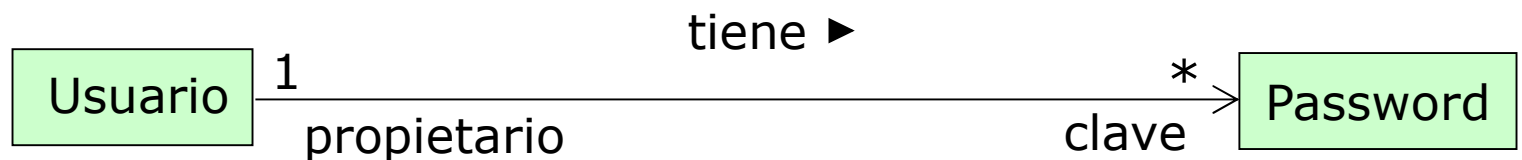


UML - Relación de **ASOCIACION**

- La **multiplicidad** es denotada por: 0..1, 1, 1..*, 5..9, * (denota 0..*), 3..*, etc.



- La **navegación** se denota con una punta de flecha en un extremo de la asociación e indica que se podrá navegar hacia los objetos de la clase apuntada pero no a la inversa.



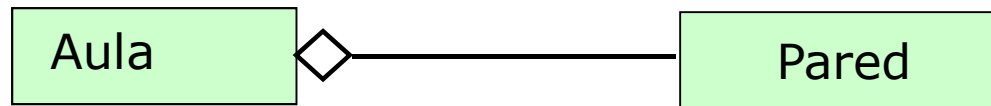
- Si la **navegación** es bidireccional la asociación no posee flechas.

UML - Relación de **AGREGACION**

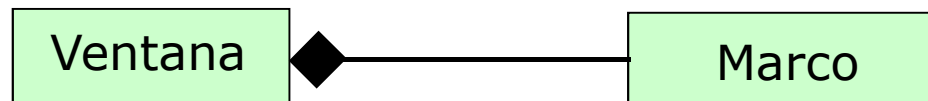
Es un tipo especial de asociación.

Representa una relación estructural entre un todo y sus partes. "tiene-un".

- **Simple**: es conceptual, solo distingue un todo de sus partes.

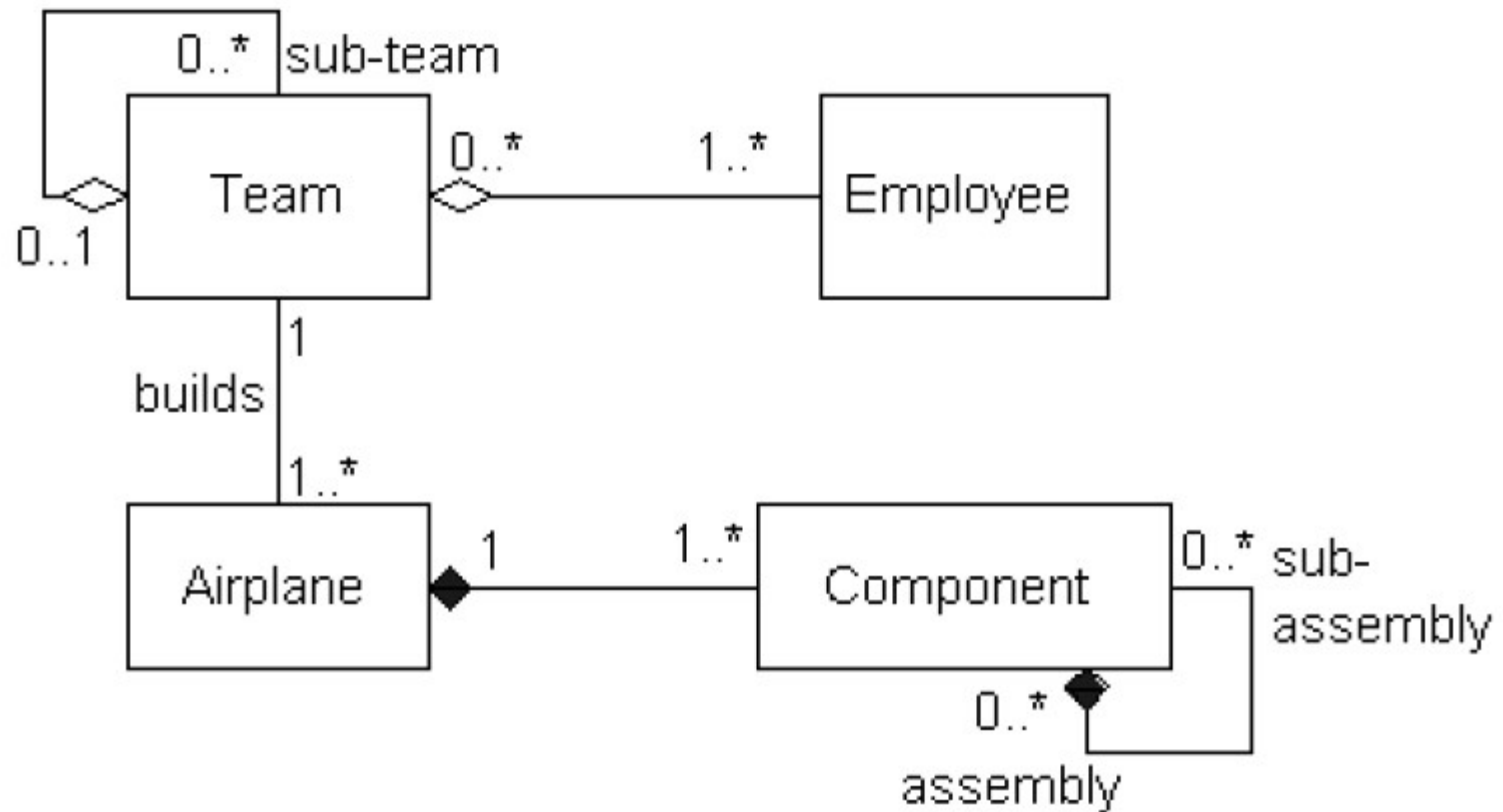


- **Compuesta**: existe una fuerte relación de pertenencia y vidas coincidentes de la parte con el todo. Habitualmente llamada **Composición**.



UML - Relación de **AGREGACION**

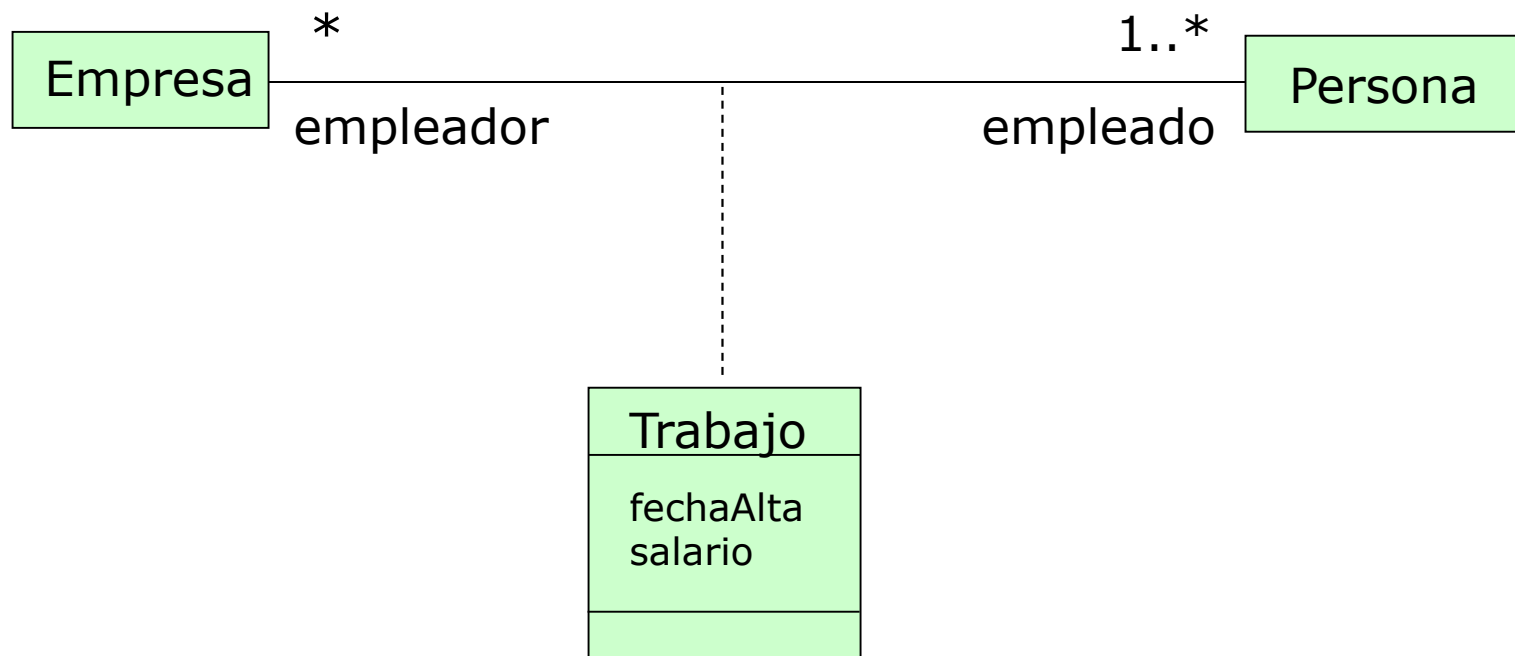
Ejemplo



UML - Relación **CLASE ASOCIACION**

Es una asociación entre dos clases, donde la propia relación tiene propiedades de asociación y de clase.

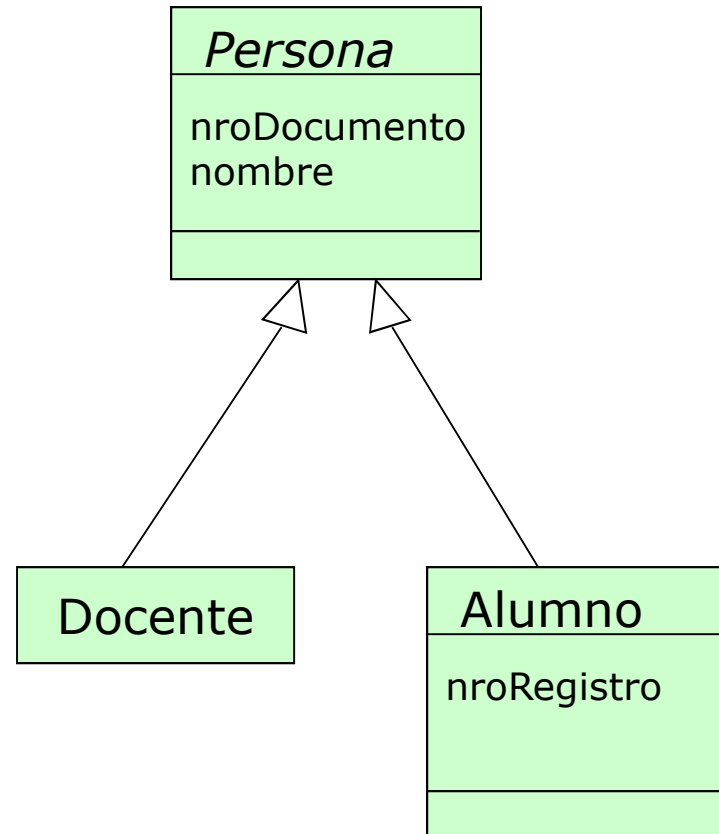
Gráficamente, se denota como una clase conectada con una línea punteada a la relación de asociación entre las dos clases involucradas.



UML - Relación de **GENERALIZACION**

Es una relación de especialización/ generalización.

El hijo comparte la estructura y el comportamiento del padre. "es-un".

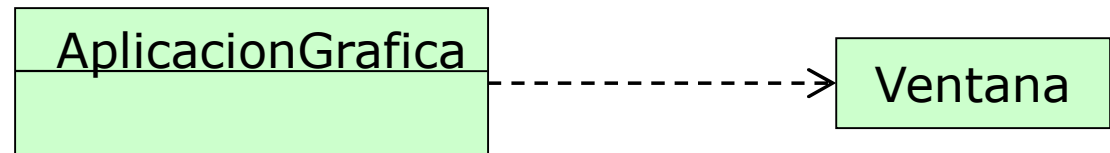


UML - Relación de **DEPENDENCIA**

Es una relación de uso entre dos elementos, la cual especifica que un cambio en un elemento (independiente) puede afectar la semántica del otro elemento (dependiente) que lo utiliza.

Gráficamente, una dependencia se denota con una flecha discontinua dirigida desde la clase dependiente a la clase de la cual depende (independiente).

Por ej: una aplicación grafica que instancia una ventana (la creación del objeto *ventana* esta condicionado a la instanciación proveniente desde el objeto *aplicacionGrafica*)

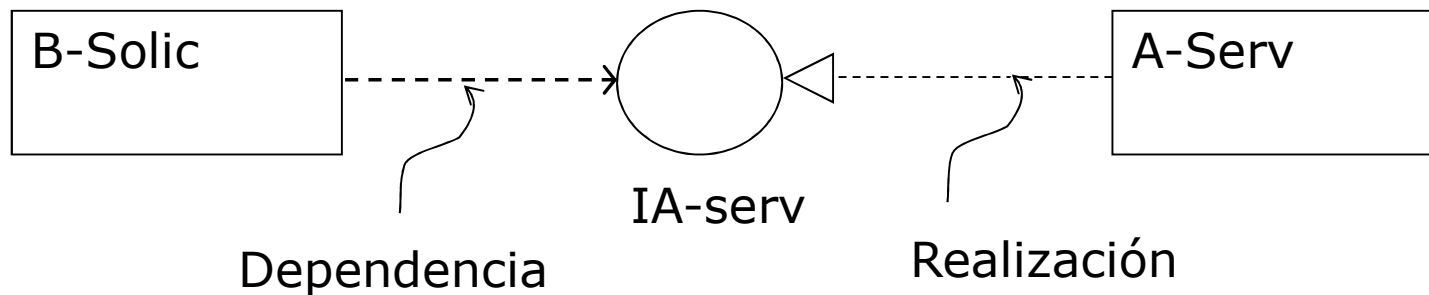


La dependencia puede contener un estereotipo para distinguir su tipo. Como `<<extend>>` e `<<include>>` para relaciones de dependencia entre casos de uso, o `<<bind>>` para las clases que especializan una clase Template.

UML - Relación de **REALIZACION**

Es una relación semántica entre clasificadores, donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.

Por ejemplo, la clase A-Serv implementa o realiza las operaciones que publica en la interface IA-Serv y la clase B-Solic usa o requiere sus servicios.

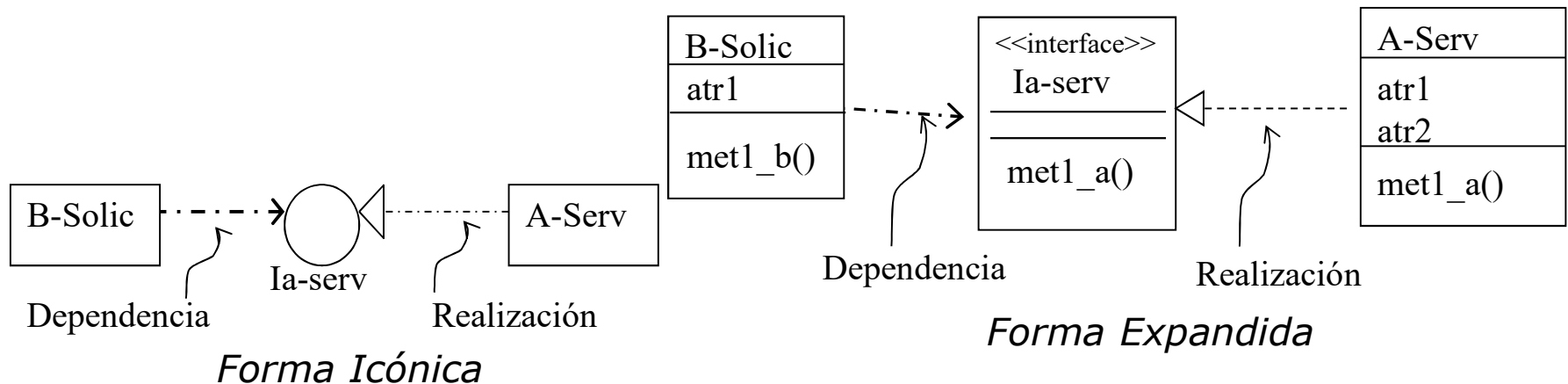


UML - Diagrama de Clases

INTERFACE

Colección de operaciones que publica los servicios de una clase o componente.

- Gralmente su nombre comienza con I, para diferenciarla de las clases.
- Se utilizan para modelar las líneas entre la especificación de lo que una abstracción hace y la implementación de cómo lo hace.
- Se conecta a la clase que la implementa con una relación de realización y a la que utiliza sus servicios con una dependencia.
- Las interfaces que una clase o componente realiza se llaman de exportación y las que usa se llaman de importación.



UML - Diagrama de Clases

USOS

➤ **Modelar el vocabulario del sistema**

Definir qué abstracciones son parte del sistema y especificarlas. Modelar las entidades más importantes. Alcanza con definir la clase, sus relaciones y sus principales atributos (sin especificar su tipo específico).

➤ **Modelar colaboraciones simples**

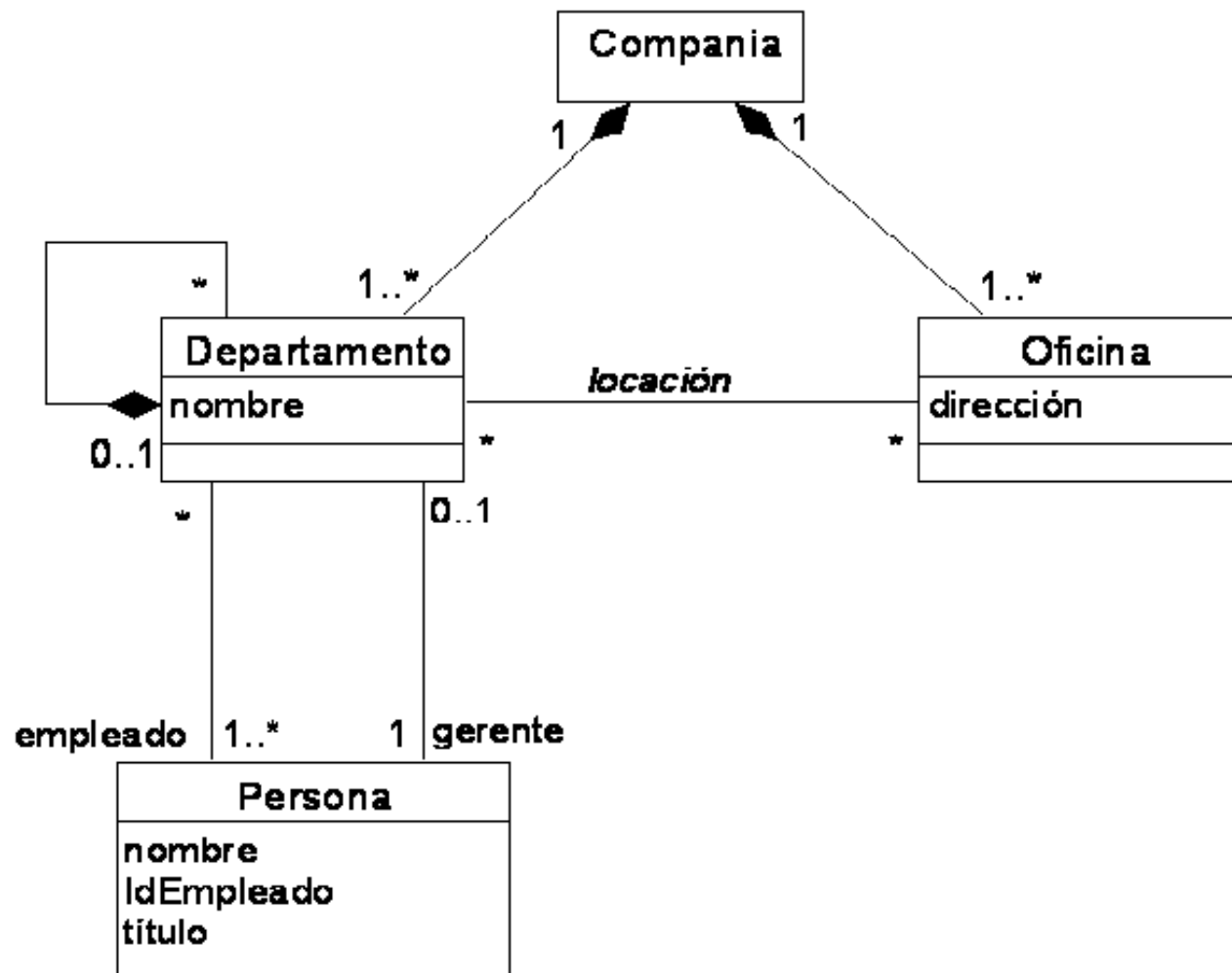
Una colaboración es una sociedad de clases, interfaces y otros elementos que trabajan juntos para proveer algún comportamiento cooperativo. Interesa modelar la clase, sus relaciones, los atributos con su tipo específico y las operaciones con su perfil.

➤ **Modelar el esquema lógico de una base de datos**

Los objetos persistentes pueden ser almacenados en una base de datos para su posterior recuperación. Las clases persistentes, sus relaciones y atributos forman las tablas de la base de datos del sistema.

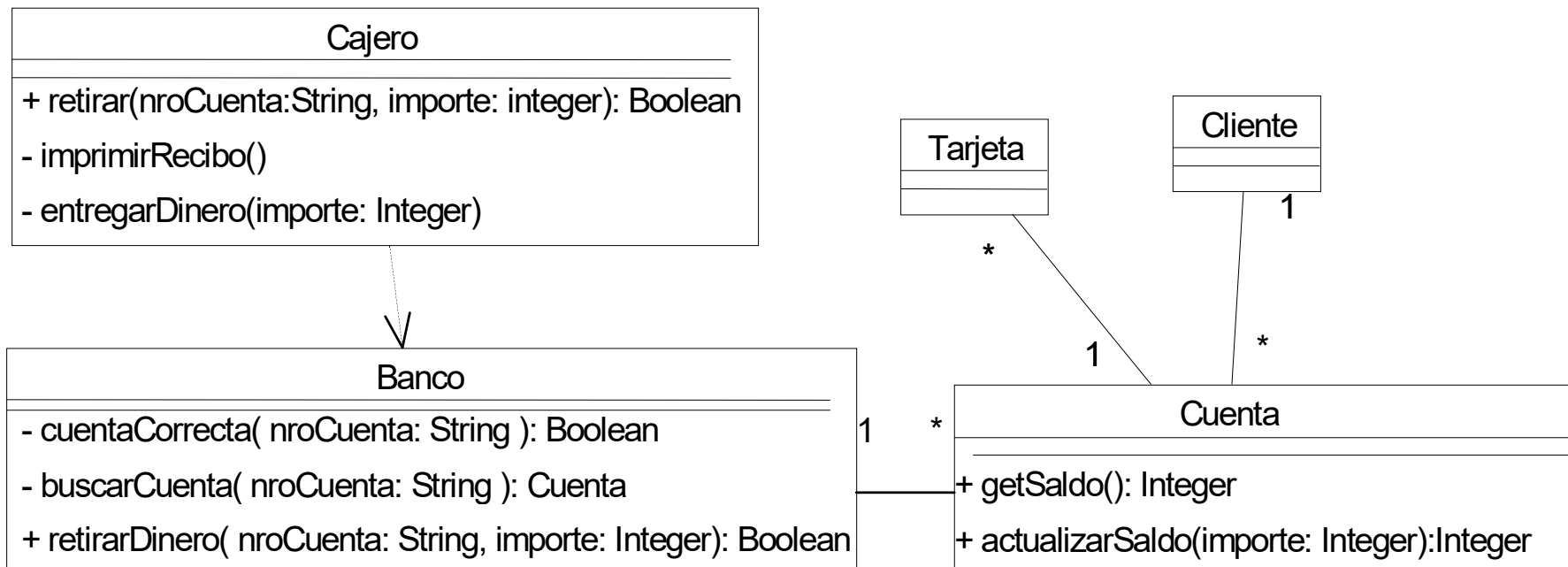
UML - Diagrama de Clases

EJEMPLO (Modela el vocabulario)



UML - Diagrama de Clases

EJEMPLO (Modela colaboraciones simples)





UML - Otros conceptos importantes

Clase Concreta: Tienen instancias directas.

Clase Abstracta: No pueden tener instancias directas. En UML, escribir su nombre en *itálica*. Ej. En la generalización, la clase padre.

Clases Hojas: Especifica que la clase no puede tener hijos. En UML, escribir la propiedad {leaf} bajo el nombre de la clase.

Clases Raíz: Especifica que la clase no puede tener padres. En UML, escribir la propiedad {root} bajo el nombre de la clase.

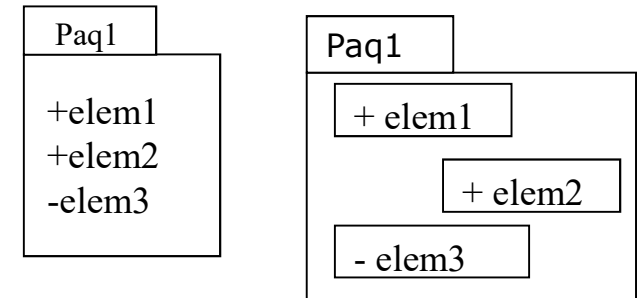
Operación Polimórfica: Operaciones con el mismo nombre en una jerarquía de clases. Las clases hijas pueden redefinir el comportamiento de operaciones de la clase padre. En tiempo de ejecución, el objeto receptor determina la elección de la operación que se ejecuta.

Operación Abstracta: sus hijos deberán implementarla. En UML, escribir en *cursiva*, ídem a las clases.

UML - Otros conceptos importantes

PAQUETE

Mecanismo de propósito general para organizar elementos en grupos.
Se representa como una carpeta.



- Puede contener clases, casos de uso, interfaces, componentes, diagramas u otros paquetes.
- Los elementos de una misma categoría deben tener nombres únicos en el mismo paquete. Ej. Si hay dos clases con igual nombre entonces son de paquetes diferentes. P1::claseA y P2::claseA.

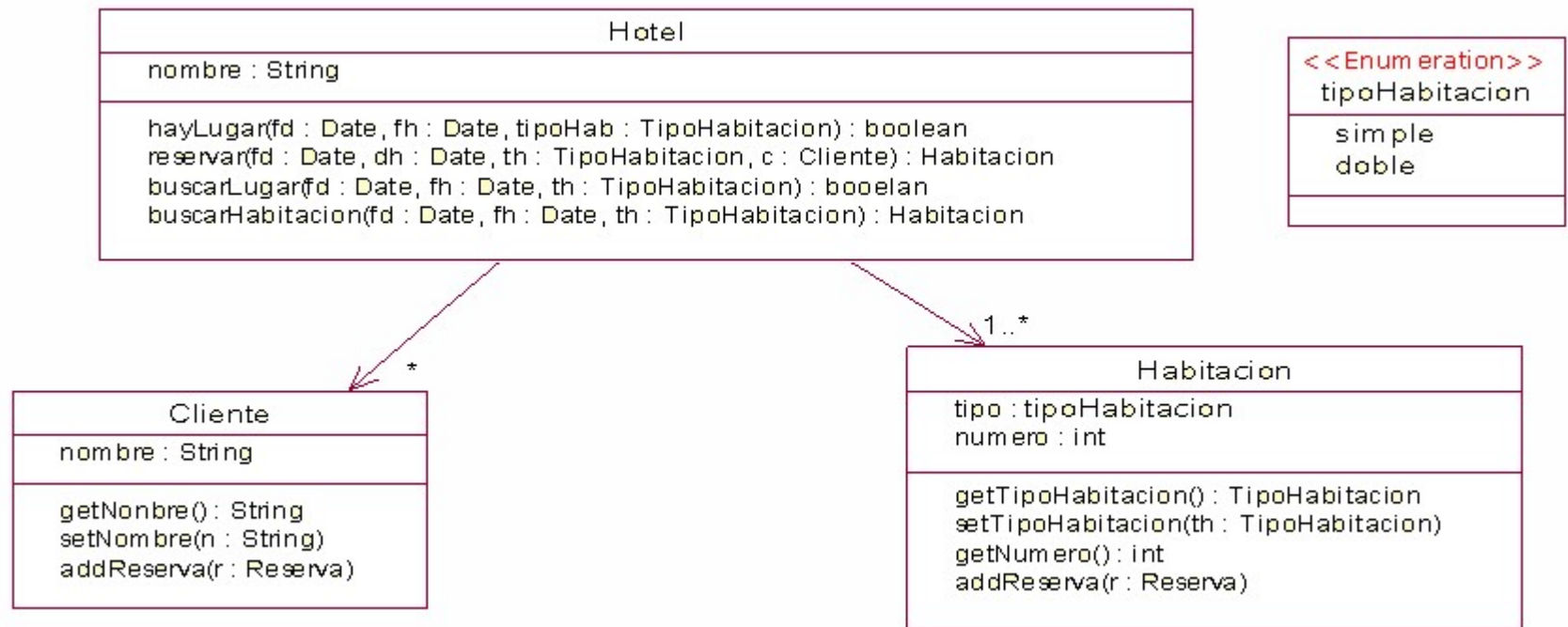
Importación: un paquete importa a otro paquete y los elementos públicos del paquete importado podrán ser utilizados por los elementos del paquete importador. En UML, se modela con una relación de dependencia adornada con el estereotipo «import».

Exportación: un paquete exporta sus elementos públicos.

UML – Clase de tipo <<Enumeration>>

<<Enumeration>>: estereotipo que permite definir una enumeración. Una enumeración es una lista de valores con nombre, utilizada como rango de valores de un tipo de atributo particular.

(Nota: Solo si se conoce TODO el rango de valores de un atributo se define como una enumeración → discusión)

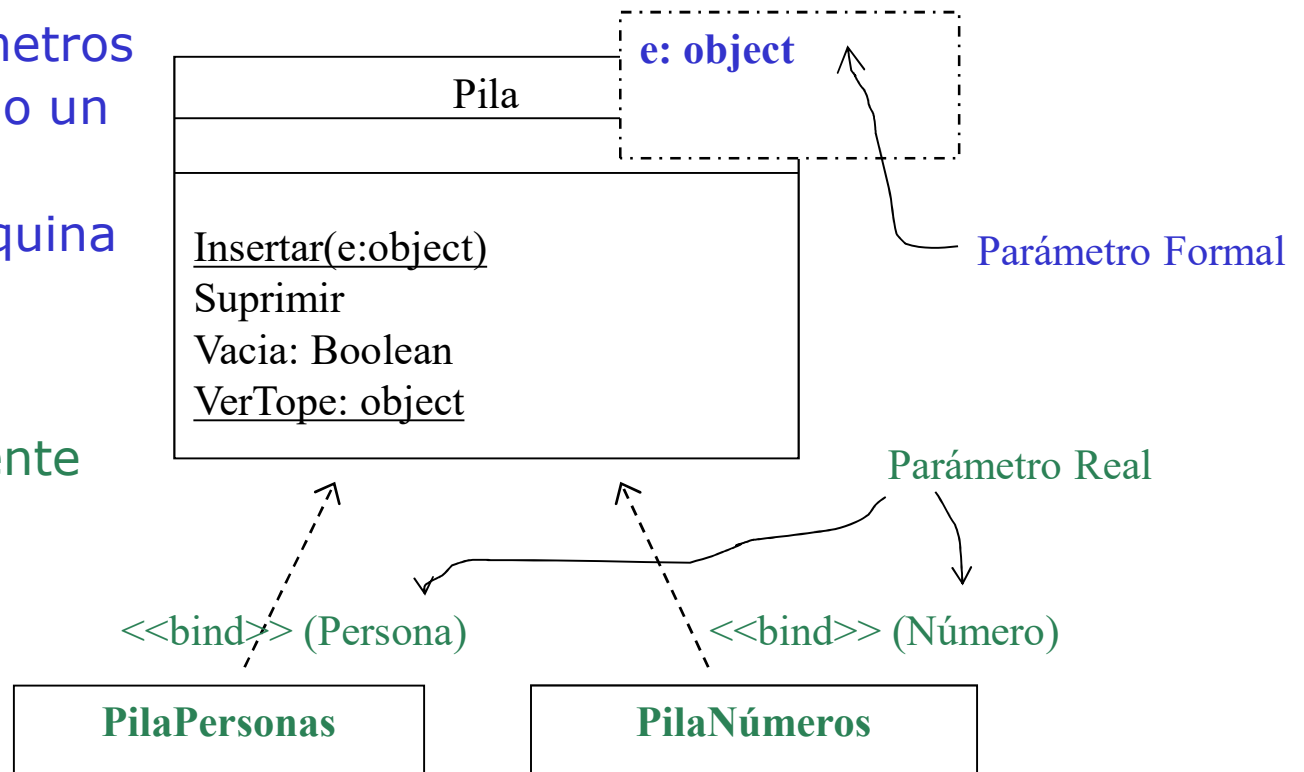


UML – Clase Template

- Representan a una familia de clases e incluyen parámetros.
- Para utilizar una clase parametrizada hay que instanciarla, es decir, ligar los parámetros formales con los parámetros reales.

➤ En UML, los parámetros se modelan utilizando un recuadro con borde discontinuo en la esquina superior derecha.

➤ Las instancias se modelan explícitamente utilizando una dependencia estereotipada, con `<<bind>>`.



Bibliografía Consultada



➤ Capítulo 1 al 12: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.

➤ <http://www.uml.org>

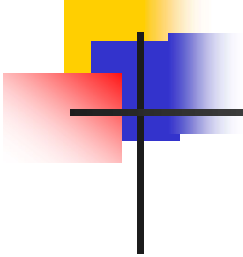


Diagrama de Objetos

UML – Instancia

- Una instancia (objeto) denota una manifestación concreta de una abstracción (clase). Se utilizan para modelar elementos concretos del mundo real.
- La abstracción a la que corresponde una instancia, especifica las características comunes a todas las instancias de su tipo.
- En UML, se representa subrayando su nombre, y dos puntos indican la abstracción a la que corresponde o que es anónima.

c1:Cliente

unCliente

Instancia u objeto con nombre

: Cliente

: Cliente

Instancia u objeto Anónima

Multiobjetos



UML – Instancia

Operaciones: Las operaciones definidas para la abstracción o clase son aplicables a las instancias particulares de la misma. Se denota como `instancia.operacion()`.

Estado de la Instancia: El estado almacena los efectos de las operaciones. Es dinámico. Es posible visualizar su estado en un momento dado de tiempo y espacio.

Con UML, se puede modelar el valor de los atributos de un objeto o instancia en un momento dado. Por ejemplo:

unCliente
nombre=Bruno fechaNac=09/06/99



UML - Diagrama de Objetos

- Permiten modelar instancias de clases.
- Modela vistas estáticas del sistema. Expresa la parte estática de una interacción.
- Modela una instantánea del sistema en un momento concreto. Modela la representación de un conjunto de objetos, sus estados y enlaces.

Elementos de un Diagrama de Objetos

- objetos (instancias de clases)
- enlaces (instancias de relaciones)



UML - Diagrama de Objetos - Usos

- **Modelado de estructuras de objetos:** Permiten mostrar significativamente conjuntos interesantes de objetos concretos, sus estados y enlaces en un momento dado.

- Resultan muy útiles para hacer **Ingeniería Inversa**, tarea común que realiza el programador, con la ayuda de herramientas, cuando se depura un sistema.

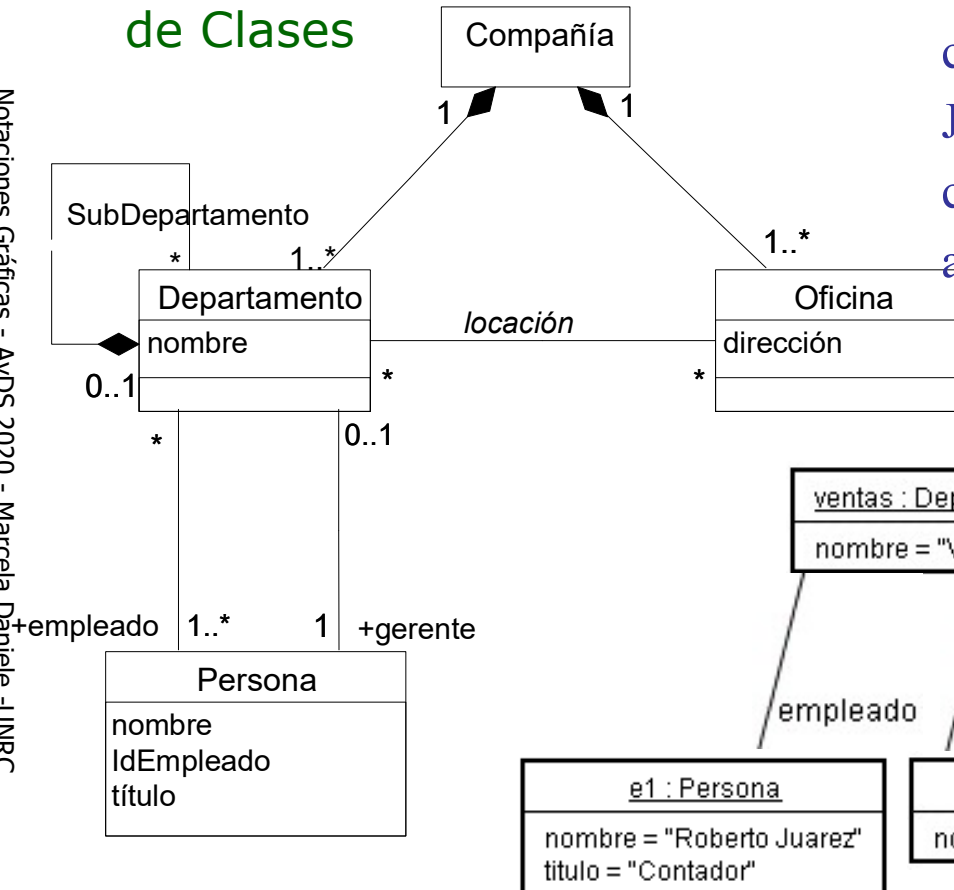
Ej: un diagrama de objetos puede ayudar a detectar dónde se invalida el estado de un objeto o su relación con otros.

Un diagrama de objetos debe contener solo aquellos elementos esenciales, es decir, solo aquellos valores de atributos y adornos esenciales para comprender el modelo específico.

UML – Ejemplo

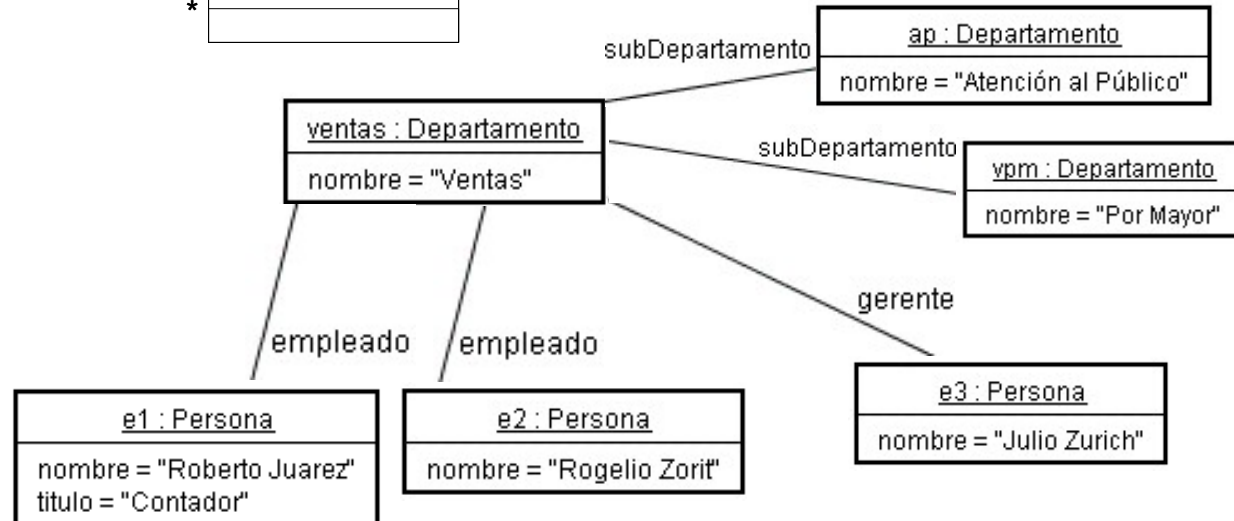
Diagrama de Objetos

Diagrama de Clases



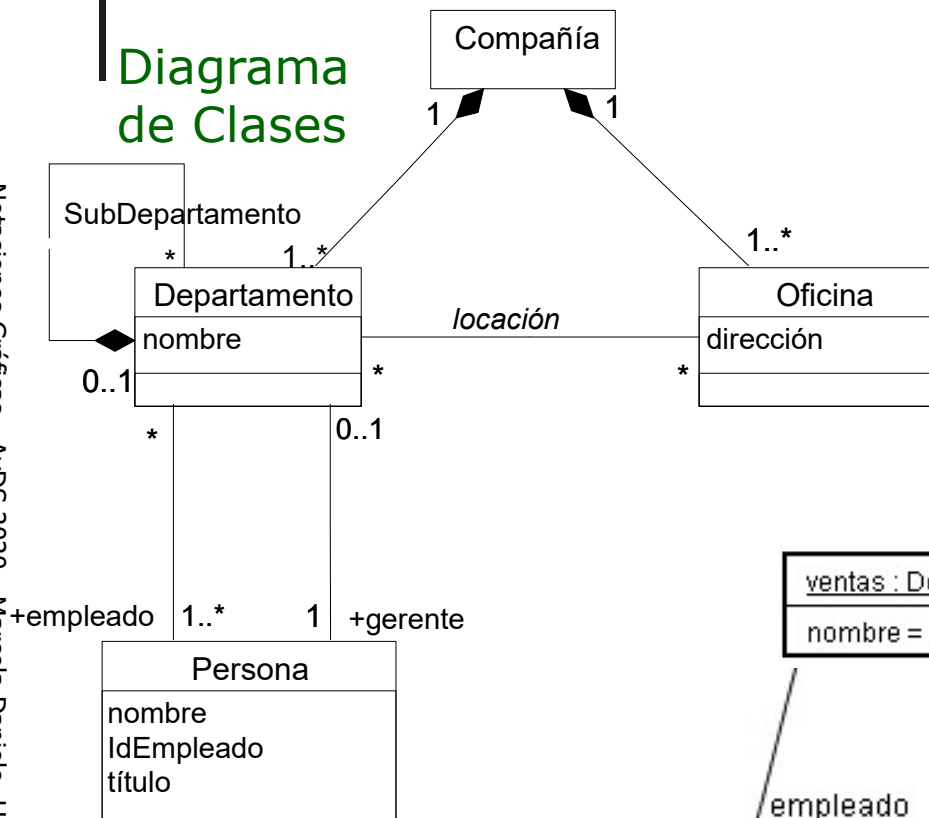
Para construir un DO necesitamos el Diagrama de Clases y un Escenario.

Escenario 1: Los nuevos empleados del dpto. de Ventas son Roberto Juarez, contador, y Rogelio Sorit. Mientras que Julio Zurich pasó a la gerencia. Además se crearon los dptos. Por Mayor y Atención al Público, ambos dependen de Ventas.

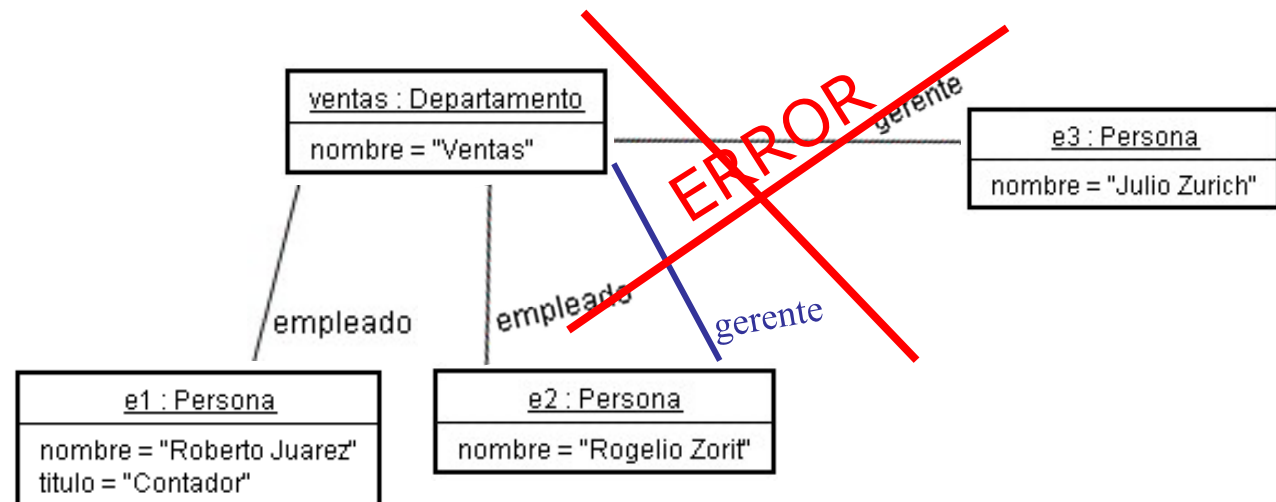


UML – Diagrama de Objetos ejemplo

Diagrama de Clases



Escenario 2: Los nuevos empleados del dpto. de Ventas son Roberto Juarez, contador, y Rogelio Sorit. En tanto que Julio Zurich y Rogelio Sorit comparten la gerencia de dicho departamento.





Bibliografía Consultada

- Capítulo 13 y 14: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.
- <http://www.uml.org>

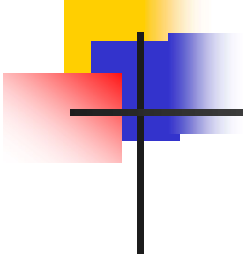


Diagrama de Actividades



UML – DIAGRAMA DE ACTIVIDADES

- Modela aspectos dinámicos de un sistema.
- Modela un flujo de control de operaciones.
- Modela el flujo de un objeto conforme pasa de estado a estado en diferentes puntos del flujo de control.
- Un DA es un tipo especial de máquina de estados.

Actividad

Es una ejecución no atómica en curso dentro de una máquina de estados, que finalmente produce alguna acción.

Acción

Es una ejecución atómica que produce un cambio en el estado del sistema o el retorno de un valor. Pueden ser: llamada a otra operación, envío de una señal, crear o destruir un objeto, un cálculo, evaluación de una expresión.



UML – DIAGRAMA DE ACTIVIDADES

ELEMENTOS DE UN DA

- Estado Inicial
- Estado Final
- Estados de Acción
- Estados de Actividad
- Transiciones
- Objetos
- Calles
- Bifurcaciones
- Divisiones
- Uniones
- Estados simples y compuestos

UML – DIAGRAMA DE ACTIVIDADES

Elementos

Estado Inicial

Indica el comienzo de un flujo de control.

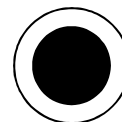
Gráficamente: un círculo relleno.



Estado Final

Indica el fin de un flujo de control.

Gráficamente: un círculo relleno dentro de una circunferencia.



UML – DIAGRAMA DE ACTIVIDADES

Elementos

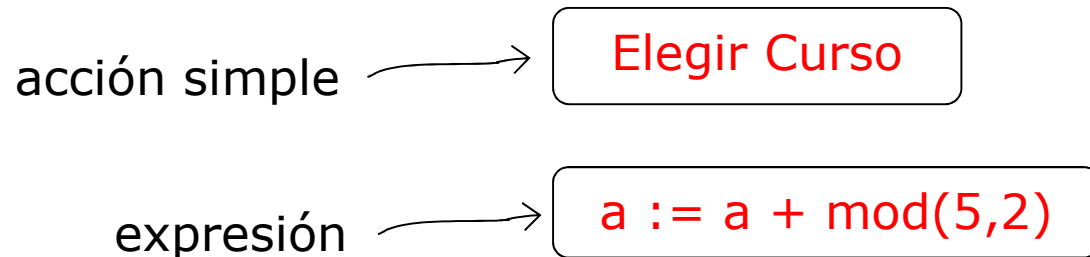
Estado de Acción

Son estados del sistema, y cada uno representa la ejecución de una acción.

No se pueden descomponer. Son atómicos, pueden ocurrir eventos pero no se interrumpe su ejecución.

Generalmente, su ejecución conlleva un tiempo insignificante.

Gráficamente: Ej:



UML – DIAGRAMA DE ACTIVIDADES

Elementos

Estado de Actividad

Pueden descomponerse en otros estados de actividad o estados de acción.

Son no atómicos, pueden ser interrumpidos por eventos.

Puede contener acciones de entrada y salida, y especificaciones de submáquinas.

Generalmente, invierten algún tiempo para completarse.

Gráficamente: Ej:

Procesar Inscripción Curso()

entry / Elegir Formulario()

exit / Entregar Recibo()

UML – DIAGRAMA DE ACTIVIDADES

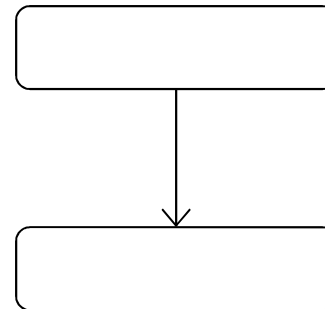
Elementos

Transiciones

Especifica el camino de un estado de actividad o de acción al siguiente.

Llamadas **transiciones sin disparadores o de terminación**, porque el control pasa inmediatamente al siguiente estado del flujo de control, una vez que se terminó la tarea del estado origen.

Gráficamente: una línea dirigida.



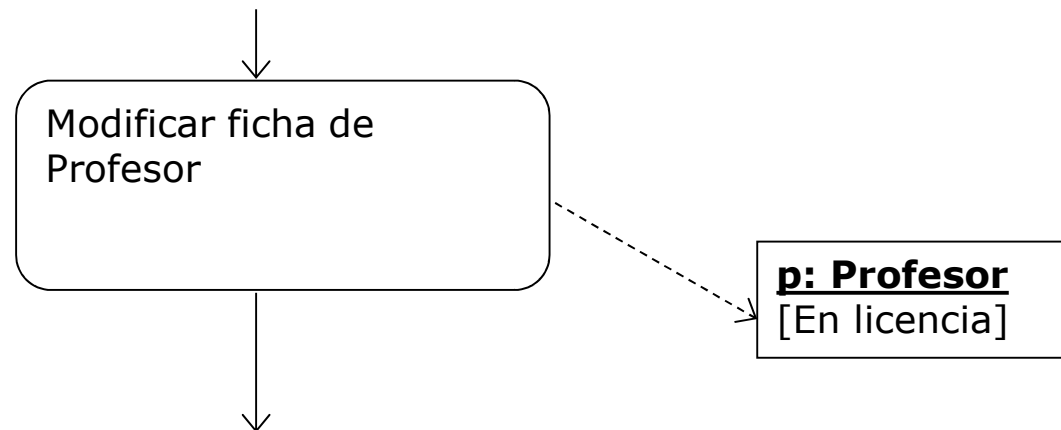
UML – DIAGRAMA DE ACTIVIDADES

Elementos

Objetos

Los objetos involucrados en un flujo de control pueden conectarse con una dependencia a la actividad o transición que los crea, destruye o modifica.

El objeto se representa con el nombre, y puede contener el valor de sus atributos y su estado.



UML – DIAGRAMA DE ACTIVIDADES

Elementos

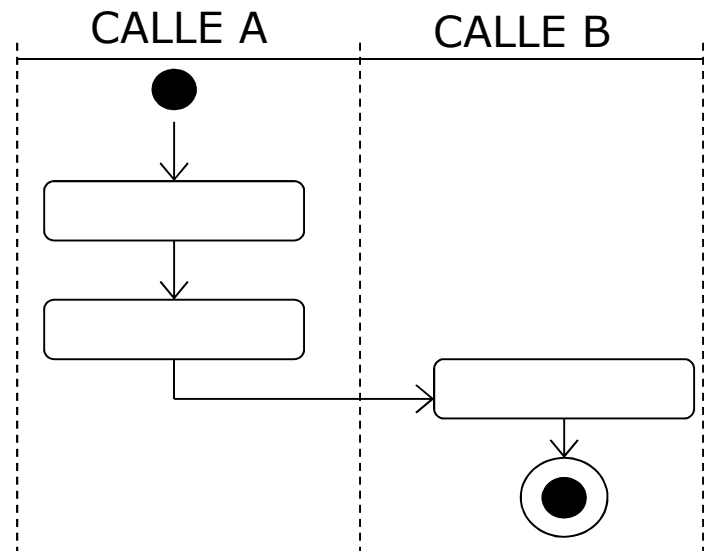
Calles (Swimlanes)

Una calle representa una unidad organizacional responsable de un grupo de actividades.

Cada actividad pertenece a una única calle.

Las transiciones pueden pasar de una calle a la otra.

Cada calle tiene un nombre.



UML – DIAGRAMA DE ACTIVIDADES

Elementos

Bifurcación

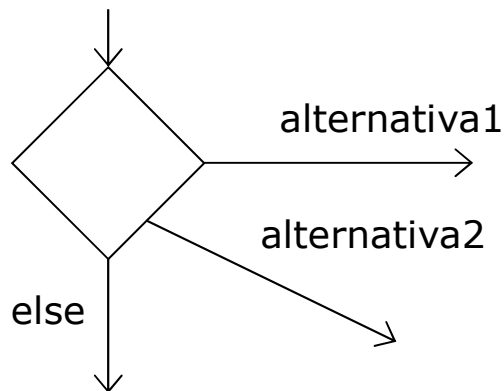
Especifica caminos alternativos.

Contiene una transición de entrada y dos o más de salida.

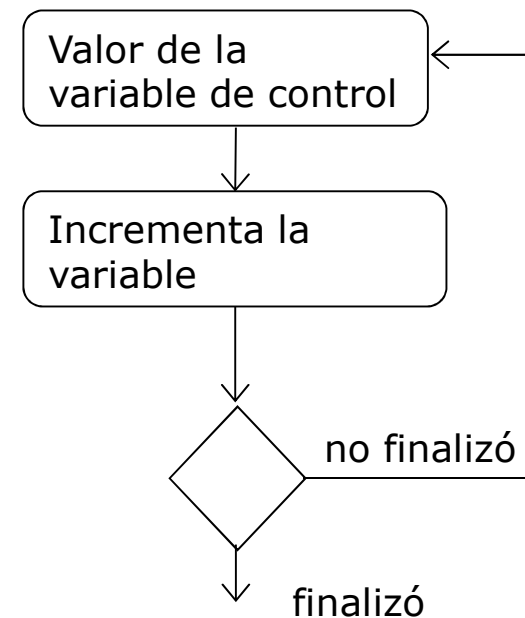
Cada transición de salida lleva una condición o expresión booleana, que no deben solaparse unas con otras.

Se puede utilizar *else*, para indicar el camino alternativo si no se cumplen los demás.

Gráficamente:



Iteración



UML – DIAGRAMA DE ACTIVIDADES

Elementos

División y Unión

Permiten modelar flujos concurrentes, se utiliza un barra ancha de sincronización, vertical u horizontal.

División

Representa la separación de un flujo de control en dos o más flujos concurrentes.

Unión

Representa la sincronización de dos o más flujos de control concurrentes, es decir, cada uno espera hasta que todos los flujos de entrada hayan alcanzado la unión, luego continúa un único flujo.

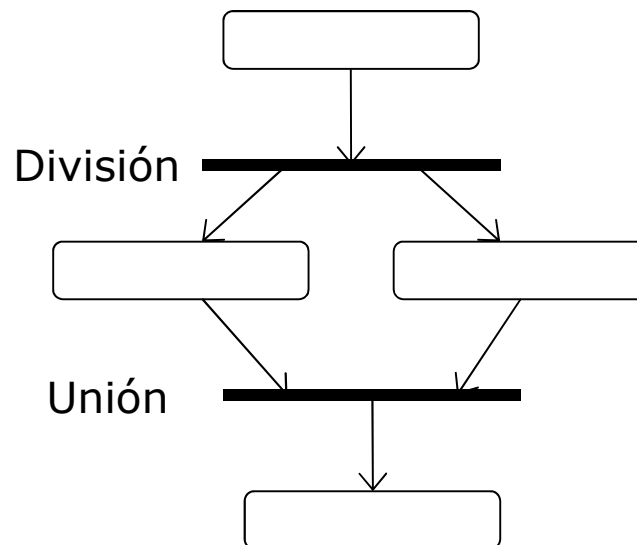
UML – DIAGRAMA DE ACTIVIDADES

Elementos

División y Unión

La **División** puede tener una transición de entrada y dos o más de salida.

La **Unión** puede tener dos o más transiciones de entrada y una de salida.



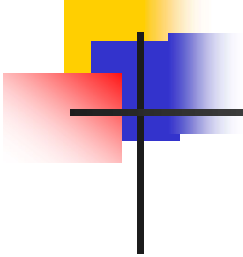


Diagrama de Estados



Eventos

Un evento es la especificación de un acontecimiento significativo que tiene lugar en el tiempo y el espacio.

Asíncronos: pueden suceder en cualquier instante.

Ej: señal, el paso del tiempo, cambio de estado.

Síncronos: representan la invocación de una operación.

Ej: llamada.

Eventos Internos: fluyen entre los objetos.

Ej: overflow.

Eventos Externos: fluyen entre el sistema y sus actores.

Ej: pulsación de un botón.

Eventos asíncronos y síncronos

Evento de Señal

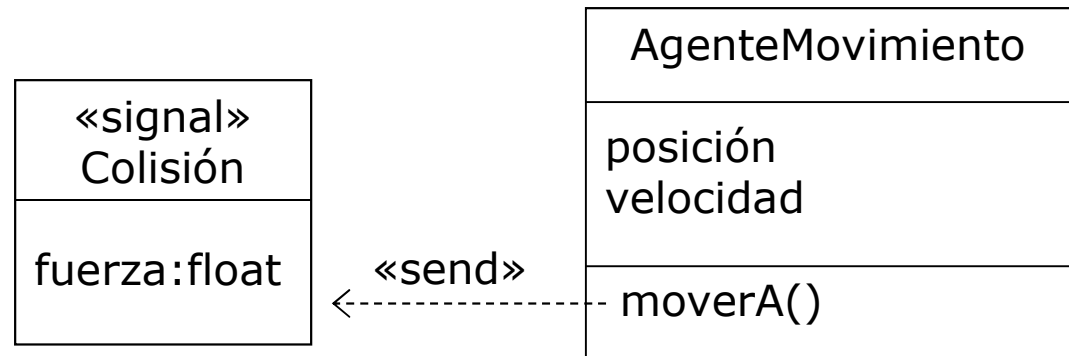
El emisor envía una señal al receptor, y sin esperar respuesta sigue su curso. No se sincronizan.

Se modelan como clases estereotipadas «signal».

Pueden contener instancias, atributos y operaciones.

Las operaciones pueden enviar señales. Se indica con una dependencia y el estereotipo «send».

Gráficamente, ej:



Las excepciones son otro tipo de señal, estereotipadas «exception».

Eventos asíncronos y síncronos

Evento de Llamada

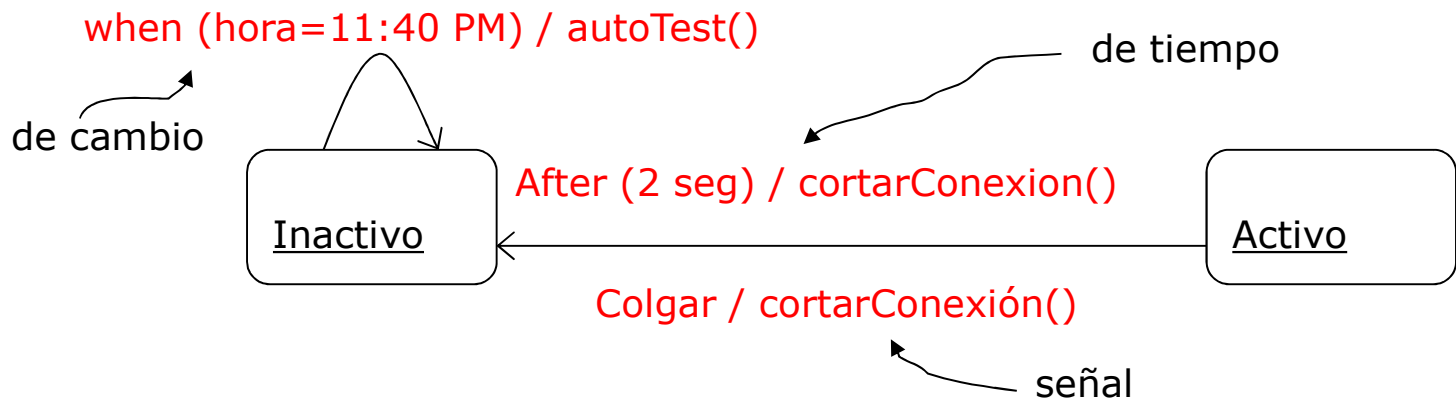
Representa la invocación de una operación. Se sincronizan el emisor y el receptor. El emisor espera respuesta.

Evento de Tiempo

Paso del tiempo. Se modela con **after** seguida de una expresión que se evalúa para producir un período de tiempo.

Evento de Cambio

Cambio de estado o cumplimiento de una condición. Se modela con **when** seguida de una expresión booleana.





Máquina de Estados

Una máquina de estados especifica la secuencia de estados por las que pasa un objeto a lo largo de su vida en respuesta a eventos.



Con UML puede visualizarse de dos maneras:

- Con un **Diagrama de Actividades**
mostrando las actividades que tienen lugar sobre el objeto
- Con un **Diagrama de Estados**
mostrando el comportamiento del objeto dirigido por eventos

UML – Diagrama de Estados

- Modela aspectos dinámicos de un sistema.
- Modela el flujo de control entre los estados de **un objeto**.
- Modela objetos reactivos, es decir, aquellos objetos que tienen un ciclo de vida bien definido, donde su comportamiento se ve afectado por el pasado y generalmente están ociosos hasta que reciben un evento.
- En UML, un Diagrama de Estados representa una máquina de estados.

ELEMENTOS DE UN DIAGRAMA DE ESTADOS

- Estado Inicial 
- Estado Final 
- Estados del objeto
- Transiciones

UML – Diagrama de Estados

Elementos - **Estado**

Estado

Situación en la vida de un objeto durante el cual satisface alguna condición, realiza alguna actividad o espera algún evento.

Puede contener: nombre, acción de entrada, acción de salida, transición interna, actividad y eventos diferidos.

Nombre

entry/acción
exit/ acción
evento [cond. de guarda]/acc1;acc2
evento/defer
do / act1;act2



UML – Diagrama de Estados

Elementos - Estado

Acción de Entrada: se ejecuta cada vez que el objeto entra al estado. **entry / acción.** (opcional)

Acción de Salida: se ejecuta cada vez que el objeto sale del estado. **exit / acción.** (opcional)

Transición Interna: se dispara la transición, se ejecuta la acción asociada y queda en el mismo estado. (No sale)

Autotransición: ejecuta el exit, sale, ejecuta la acción, entra, ejecuta el entry, y queda en el mismo estado.

Eventos Diferidos: lista de eventos que esperan para ser disparados en otro estado del objeto. **evento / defer.**

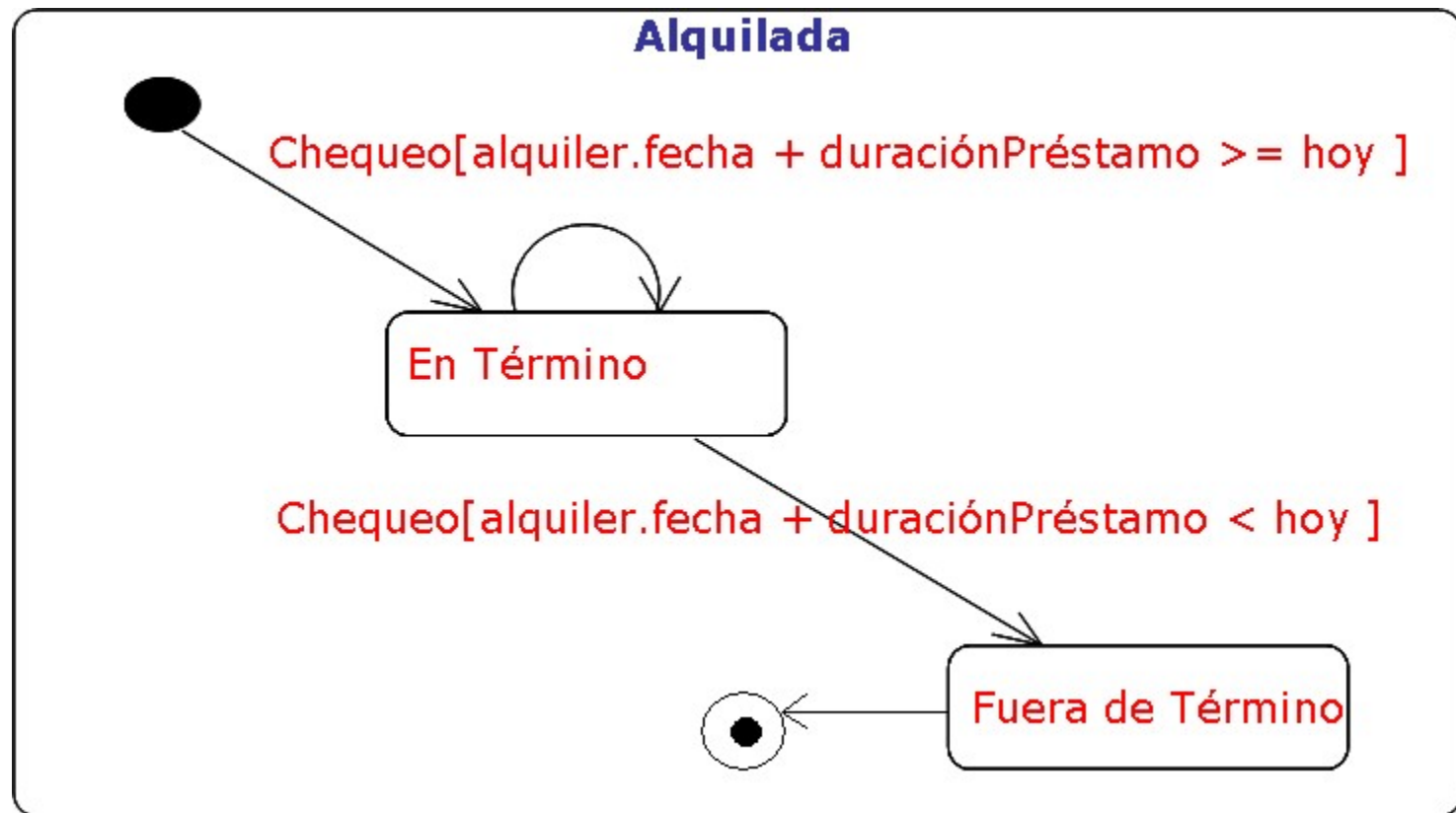
Actividad: ocurre mientras que el objeto permanece en el mismo estado. **do / act1;act2.**

UML – Diagrama de Estados

Elementos - Estado

Estado Compuesto: el estado de un objeto puede estar compuesto de otros estados.

Ej:



UML – Diagrama de Estados

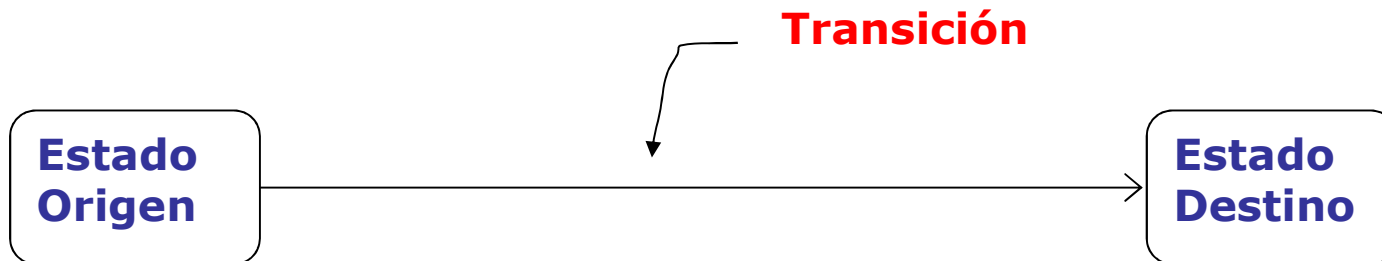
Elementos -**Transición**

Transición

Relación entre un estado origen y un estado destino. Cuando la transición es disparada el objeto pasa de un estado a otro.

Estado Origen: cuando ocurre un evento de disparo y el objeto se retira del estado. Además, si existe una condición de guarda ésta se satisface.

Estado Destino: estado activo luego que ha terminado la transición.



UML – Diagrama de Estados

Elementos -Transición

Elementos de una Transición

- **Evento de Disparo**: evento que provoca el disparo de la transición si la condición de guarda se satisface.
- **Condición de Guarda**: expresión booleana que se evalúa cuando la transición se activa por la recepción de un evento.
- **Acción**: computación atómica ejecutable que actúa sobre el objeto representado o sobre otros objetos visibles a él.





Bibliografía Consultada

- Capítulos 20 al 25: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.
- <http://www.uml.org>

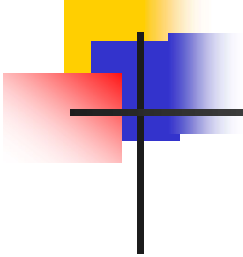


Diagrama de Casos de Uso



UML – Casos de Uso

- **Caso de Uso**: descripción de una secuencia de acciones que produce un resultado observable y de valor para un **actor** particular.
 - Una secuencia de acciones representa una interacción entre elementos externos (actores) y el sistema.
 - Requisito funcional del sistema.
- **Actor**: cualquier elemento externo al sistema que requiera el uso del mismo. Ej: una persona, otro sistema, un dispositivo externo.



UML – Casos de Uso

- Los casos de uso describen “**QUE**” hace el sistema.
- Su comportamiento se especifica a través de una descripción textual que puede contener:
 - Un nombre único
 - Pre y Pos condiciones
 - Flujo de Eventos Principal
 - Flujo de Eventos Alternativo
- Se definen escenarios particulares como instancias de un caso de uso.
- Los casos de uso se pueden organizar en paquetes.



UML – Diagrama de Casos de Uso

- Modela las funcionalidades del sistema. Modela la vista de Casos de Uso del sistema.
- Modela la interacción entre casos de uso (funcionalidades) y actores del sistema.
- Organiza los diferentes comportamientos de un sistema.
- Los elementos de un diagrama de casos de uso pueden ser:
 - Casos de Uso
 - Actores
 - Relaciones
 - Notas

UML – Diagrama de Casos de Uso

- Un **Caso de Uso** se representa gráficamente por una elipse de borde continuo incluyendo su nombre.



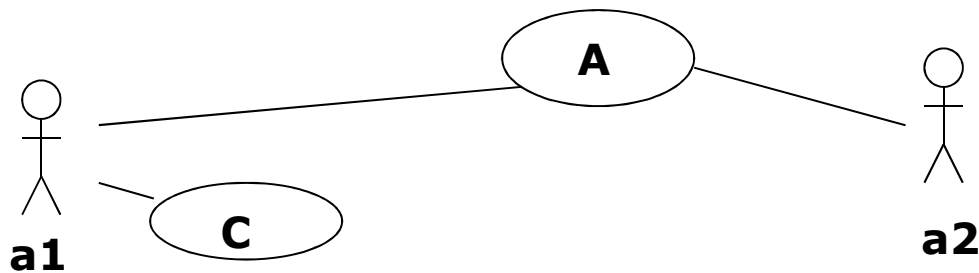
- Un **Actor** también tiene una representación e incluye un nombre que lo identifica.



UML – Diagrama de Casos de Uso

RELACIONES

- Un Caso de Uso puede asociarse con uno o mas Actores.
Se modela con relación de asociación.

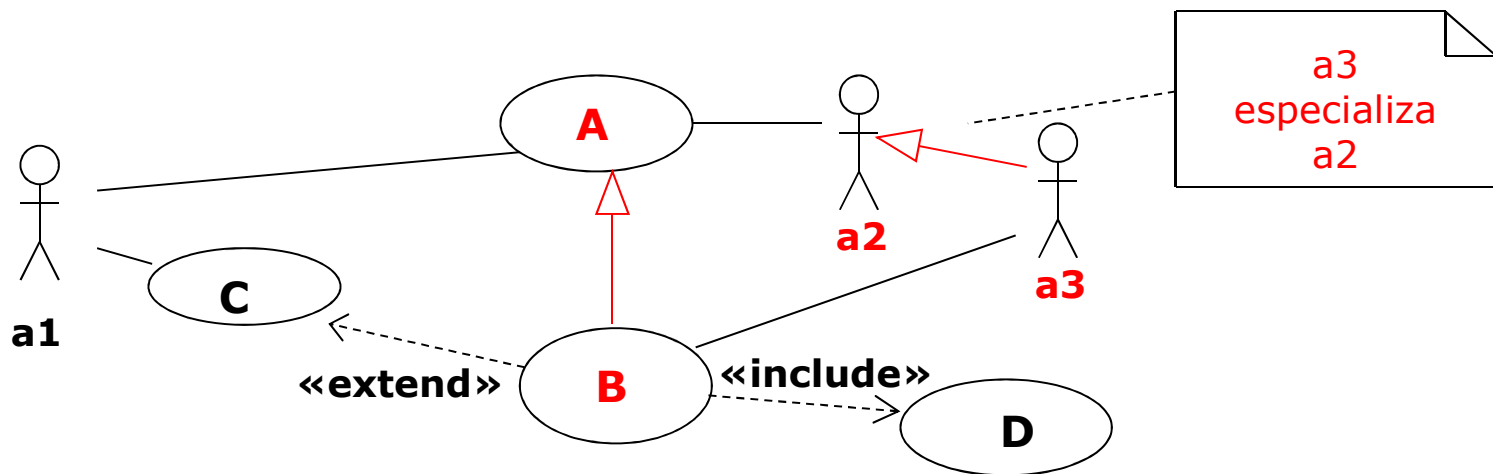


- Un Caso de Uso puede relacionarse con otros Casos de Uso.
Las relaciones posibles son: **include**, **extend** y **generalización**.
- Un actor puede generalizar o especializar a otro actor.

UML – Diagrama de Casos de Uso

➤ Relación de **Generalización**

- Las instancias de los **Casos de Uso** generalizados pueden ejecutar el comportamiento descrito en el caso de uso generalizador. **Ej: B especializa a A.**
- La relación de generalización también puede darse entre **Actores**.



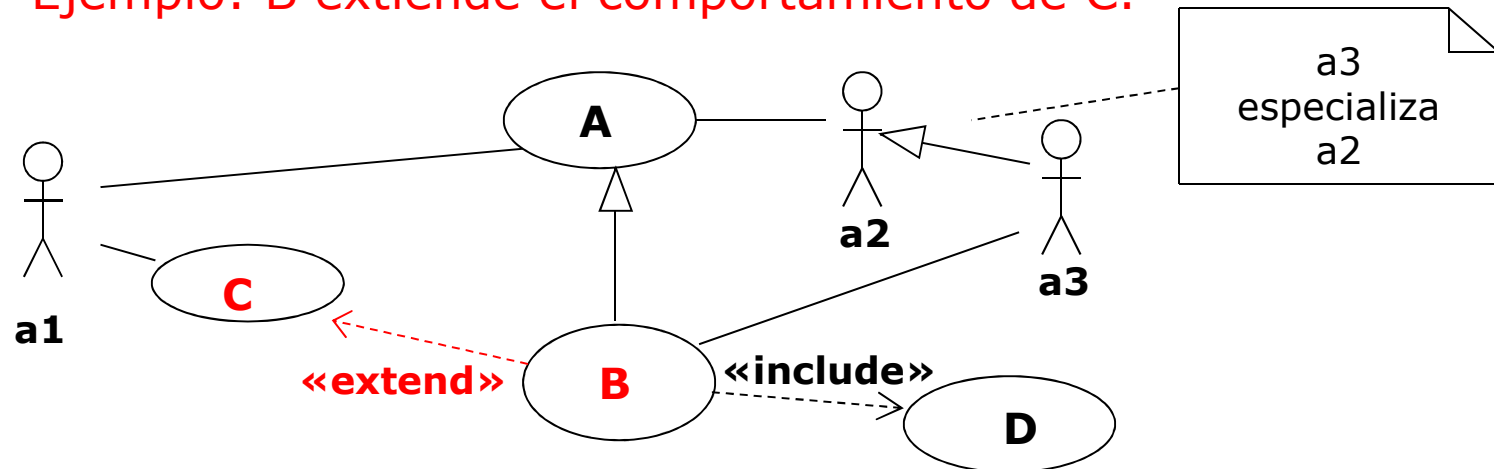
UML – Diagrama de Casos de Uso

➤ Relación de **Extensión («extend»)**: modela la posible adición de una secuencia de acciones a un caso de uso.

➤ Añade algún comportamiento extra a la descripción original de un caso de uso.

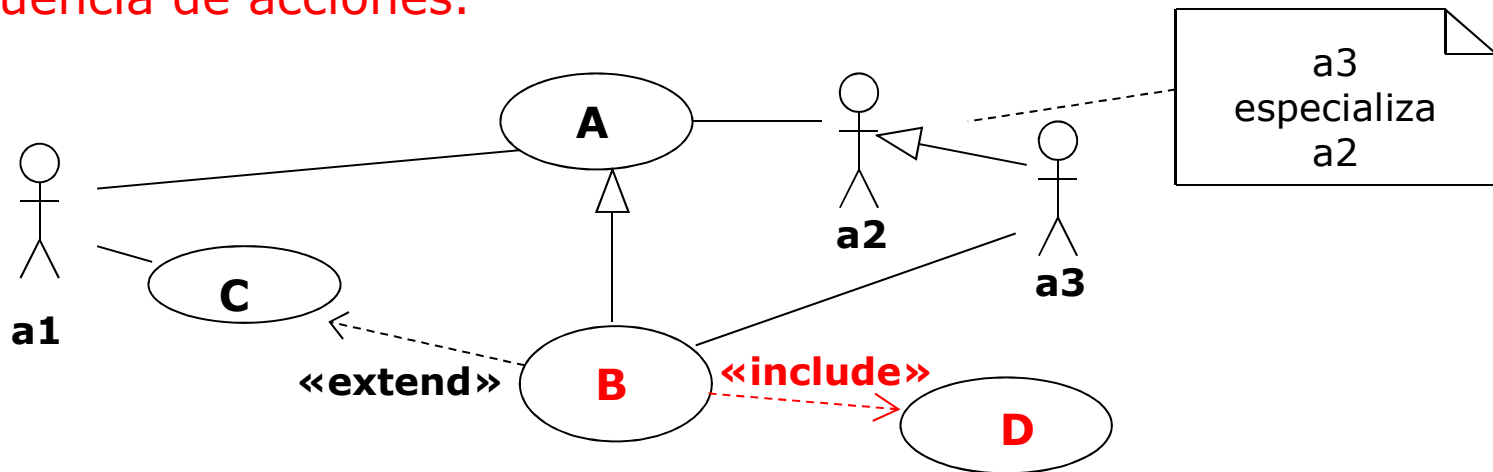
➤ Se modela con una relación de dependencia desde el Caso de Uso que extiende (B) al Caso de Uso Base (extendido-C).

➤ **Ejemplo: B extiende el comportamiento de C.**



UML – Diagrama de Casos de Uso

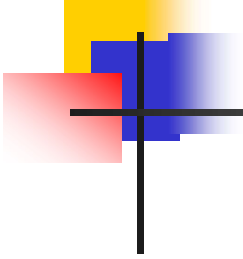
- Relación de **Inclusión («include»)**: el caso de uso origen incorpora explícitamente comportamiento de otro caso de uso en la posición especificada por el origen.
- Se modela con una relación de dependencia desde el Caso de Uso Base (**B**) al Caso de Uso incluido (**D**).
- Ejemplo: B incluye el comportamiento de D en algún punto de su secuencia de acciones.





Bibliografía Consultada

- Capítulo 17 y 18: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.
- <http://www.uml.org>



Diagramas de Interacción

- Diagrama de Comunicación
- Diagrama de Secuencia

UML –Interacciones

- Los objetos interactúan entre sí pasándose **mensajes**.
- Los objetos se conectan a través de **enlaces**.

Mensaje: especifica transmisión de información entre objetos.

Enlace: especifica un camino a lo largo del cual un objeto puede enviar un mensaje a otro objeto.

- Es una conexión semántica entre objetos.
- Es una instancia de una relación.
- Puede contener los adornos de la relación.

Interacción: conjunto de **mensajes** que se intercambian entre un conjunto de **objetos** dentro de un contexto y modelan un escenario particular.

Las **Interacciones** modelan aspectos dinámicos del sistema.

UML – Mensaje

- Especifica una transmisión de información entre objetos.
- Tiene un **emisor**, un **receptor** y una **acción**.
- **Acciones**: llamada, señal, creación y destrucción de objetos.

➤ Llamada

Invoca una operación sobre un objeto. Puede ser a sí mismo.

➤ Retorno

El **receptor** de una **llamada** devuelve un valor al **emisor**, solo si es necesario.

➤ Envío

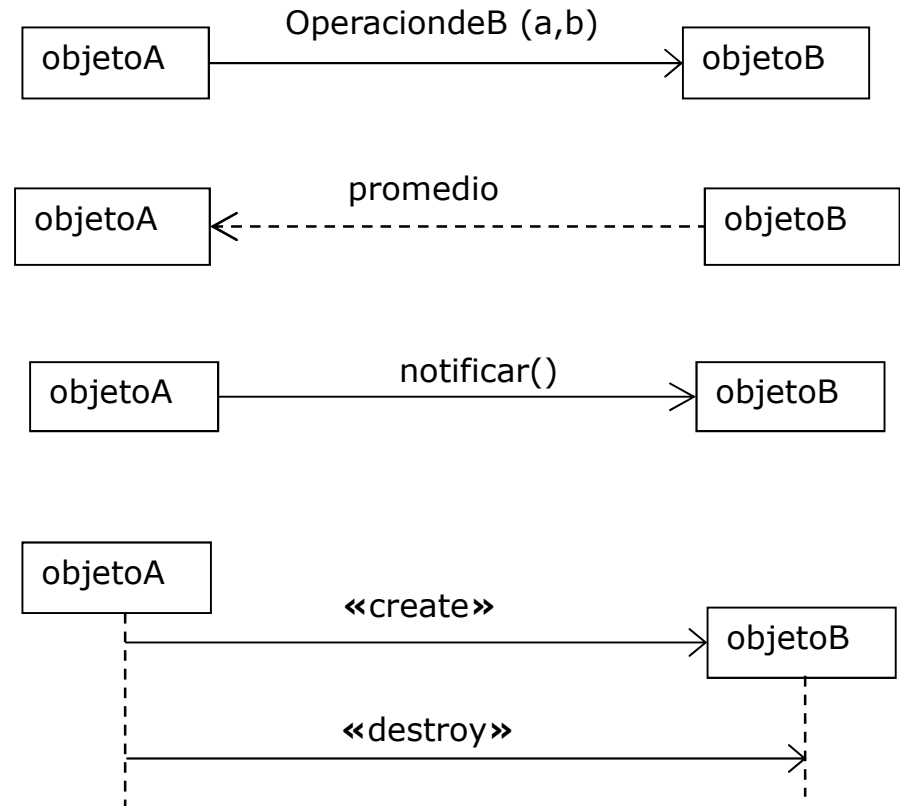
Envía una señal a un objeto.

➤ Creación

Para crear un objeto.

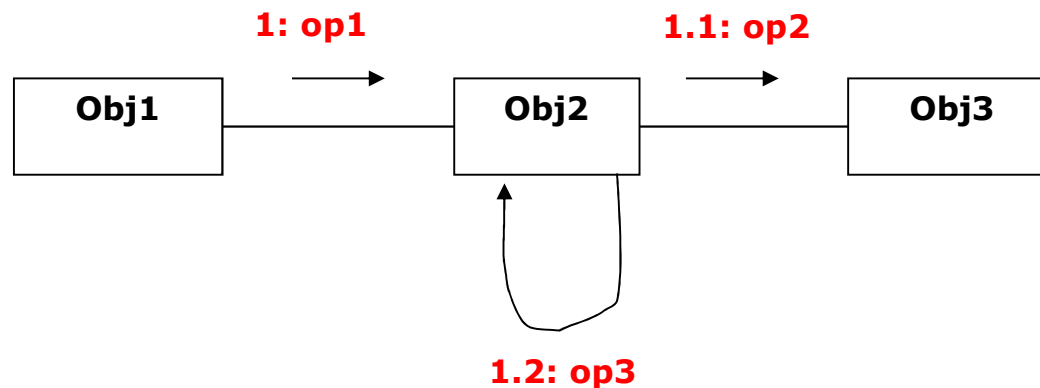
➤ Destrucción

Para destruir un objeto. Puede destruirse a sí mismo.



UML – Secuenciación

- El flujo de mensajes forma una **secuencia**.
- La secuencia es indicada por un número antes del mensaje y una flecha dirigida.
- Para modelar caminos alternativos, se coloca el mismo número de secuencia seguido de un número de subsecuencia.



Parámetros Reales

Se pueden modelar los parámetros reales enviados y también los retornos. Ej: 1.2.1: x:=operación('m')

Visualizar una interacción con UML

- Los Diagramas de Interacción permiten modelar interacciones entre objetos.
- Permiten modelar los aspectos dinámicos de un sistema.
- Se construyen representaciones gráficas a partir de escenarios que impliquen la interacción entre objetos.
- Los diagramas de interacción de UML son:
 - **Diagrama de Comunicación**
 - **Diagrama de Secuencia**

Los Diagramas de Comunicación y de Secuencia son semánticamente equivalentes.

Difieren en la forma de mostrar la interacción.

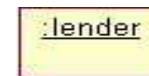
UML – Diagrama de Comunicación

Destaca la organización estructural de los objetos.

Elementos de un Diagrama de Comunicación

- **Objetos o Roles:** nodos del grafo.

Un objeto se representa gráficamente por:



O pueden estar estereotipados:



- **Enlaces o comunicaciones:** arcos del grafo.
- **Mensajes:** llevan número de secuencia y flecha dirigida.
- **Anidamiento:** se utiliza numeración decimal de Denwey. Ej: 1, 1.1, 1.1.1
- **Iteración:** colocar un * antes del número de secuencia y una cláusula de condición, si es necesario. ej. *[x>0].
- **Bifurcación:** los caminos alternativos tienen el mismo nro de secuencia, seguido del nro de subsecuencia y se distinguen por una condición.

Nota: las condiciones se indican usando: [], algún pseudocódigo o la sintaxis del lenguaje de programación.

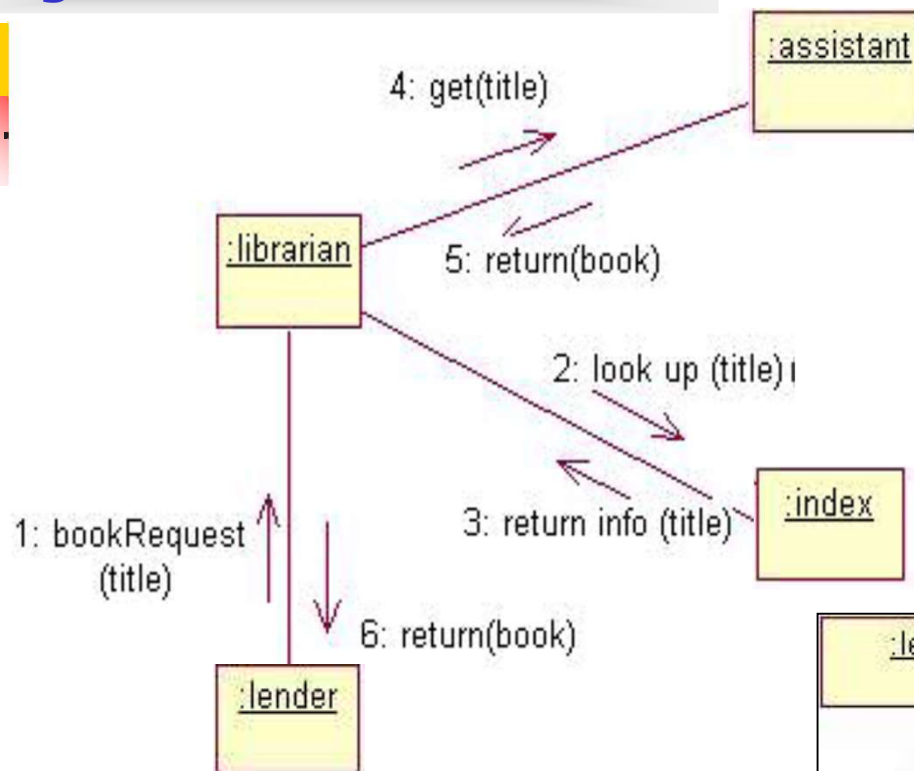
UML – Diagrama de Secuencia

Destaca el orden temporal de los mensajes.

Elementos de un Diagrama de Secuencia

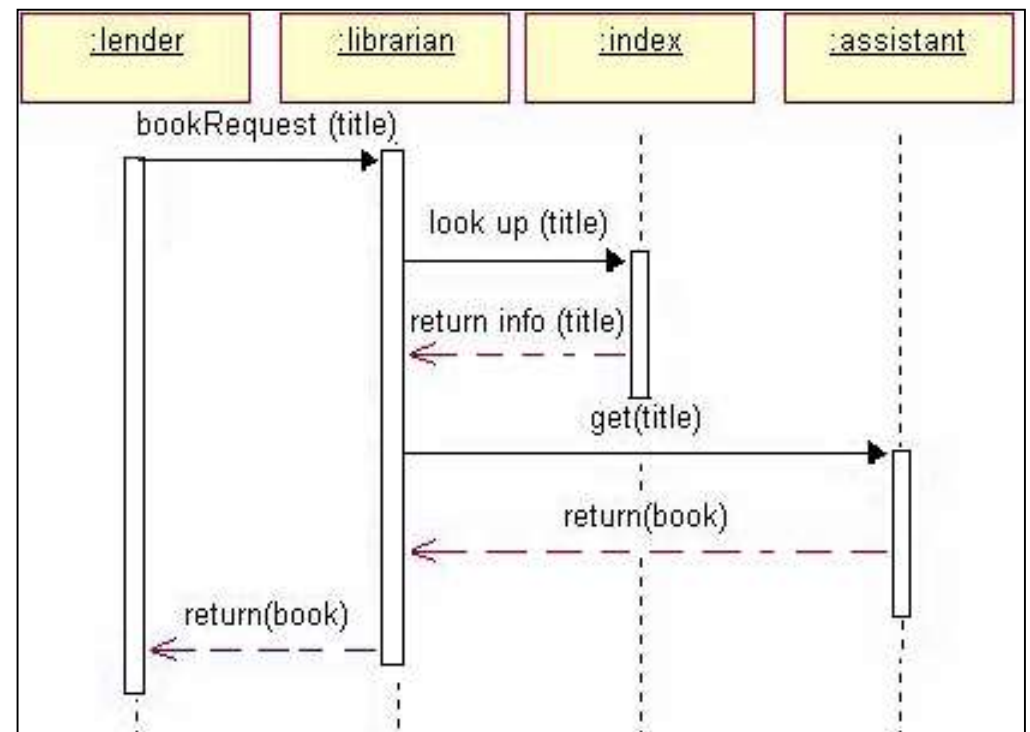
- **Objetos:** se colocan en el eje X. Generalmente, el objeto que inicia la interacción se coloca a la izquierda.
- **Mensajes:** se colocan sobre el eje Y, en el orden de sucesión en el tiempo de arriba hacia abajo.
- **Línea de Vida:** línea discontinua que representa la existencia de un objeto a lo largo de un período de tiempo.
- **Foco de Control:** representa el período de tiempo durante el cual un objeto ejecuta una acción.
- **Anidamiento:** se muestra con otro foco ligeramente a la derecha del foco padre.
- **Iteración:** se coloca un * y la condición, antes del mensaje que inicia la iteración. Ej. *[x>0].

Diagrama de Comunicación



Es necesario disponer del diagrama de clases y el escenario a representar.

Diagrama de Secuencia



Escenario:

Un lector solicita un libro al bibliotecario, y le brinda su título. El bibliotecario busca el libro en un índice y solicita al asistente que le alcance el libro.

Bibliografía Consultada



- Capítulos 16 y 19: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.
- <http://www.uml.org>



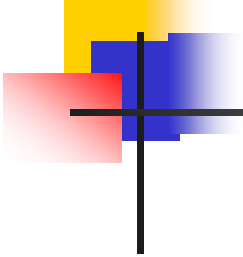
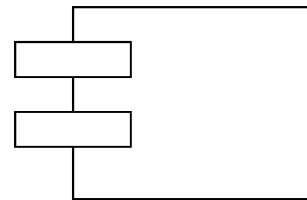


Diagrama de Componentes

UML – COMPONENTE

Un componente es una parte física y reemplazable de un sistema, conforma con un conjunto de interfaces y realiza esas interfaces.

Gráficamente en UML:



- Un componente debe tener un nombre: **simple**, ej. **cliente.java** o **de camino**, cuando está incluido en un paquete. ej. **system::dialog.dll**
- Un componente puede contener adornos, valores etiquetados e información adicional. Ej. referencia a las interfaces que realiza.



UML – COMPONENTE

- Un **componente** posee características similares a una **clase**: tiene nombre, realiza interfaces, puede participar de relaciones, puede tener instancias, puede participar en interacciones.

Porqué se diferencian?

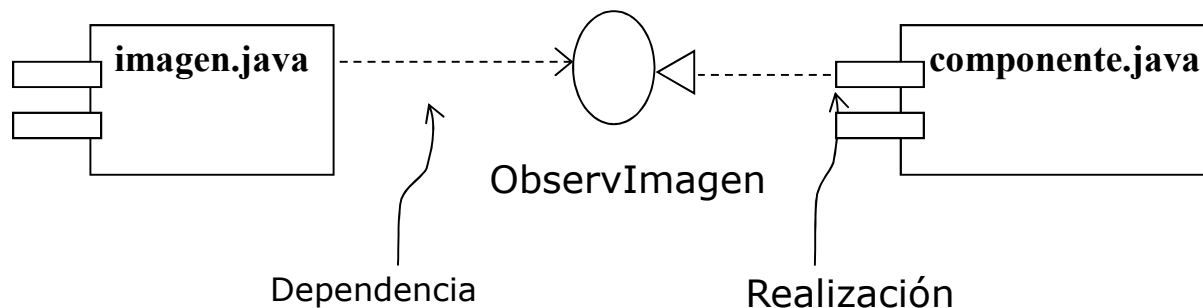
- Un **componente** representa un elemento físico (bits). Una **clase** es una abstracción lógica.
- El **componente** se puede representar en nodos físicos, la **clase** no.
- Las operaciones de un **componente** solo se alcanzan a través de interfaces. Las de una **clase** podrían ser accesibles directamente.

UML –Componentes e Interfaces

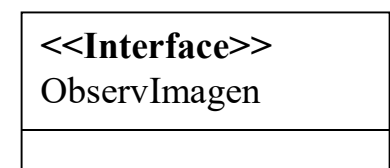
- Una interfaz contiene una colección de operaciones y se utiliza para especificar los servicios de una clase o de un componente.
- Una interfaz se conecta al componente que la implementa a través de una relación de realización, y al componente que utiliza sus servicios con una dependencia.

Gráficamente:

Forma icónica



Forma expandida





UML –Componentes e Interfaces

- **Interfaz de Exportación:** interfaz realizada por un componente, servicio que ofrece a otros componentes.
- **Interfaz de Importación:** interfaz usada por un componente.

Las interfaces permiten romper las dependencias directas entre componentes.

Un componente que usa una interfaz puede funcionar adecuadamente independientemente del componente que la realiza.



Características de un Componente

➤ **Un componente es físico**

existe en el mundo de los bits.

➤ **Un componente es reemplazable**

es posible reemplazar un componente por otro que conforme con las mismas interfaces.

➤ **Un componente es una parte de un sistema**

representa un bloque de construcción fundamental sobre el cual se puede diseñar y construir sistemas. Un sistema puede ser solo un componente en un nivel de abstracción mayor, compuesto por componentes.



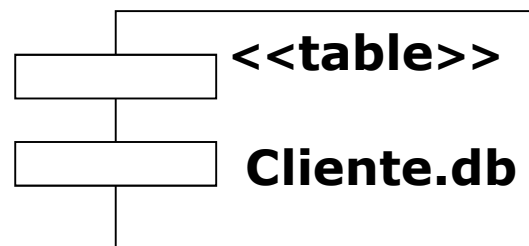
UML – Tipos de Componentes

- **Componentes de despliegue:** necesarios y suficientes para formar un sistema ejecutable. Por ejemplo: bibliotecas dinámicas (dll), ejecutables (exe).
- **Componentes productos del trabajo:** surgen durante el proceso de desarrollo y quedan al final del mismo. Por ejemplo: buscarCliente.jar, cliente.db, ManualDeUsuario.pdf.
- **Componentes de ejecución:** se crean como consecuencia de un sistema en ejecución. Por ejemplo: objetos que se instancian a partir de una dll.

Estereotipos Estándar de Componentes

- **executable**: especifica un componente ejecutable en un nodo.
- **library**: especifica una biblioteca de objetos.
- **table**: especifica una tabla de una BD.
- **file**: especifica un componente que contiene un documento con código fuente o datos.
- **document**: especifica un componente que representa un documento.

Gráficamente:





UML – Diagrama de Componentes

- Modela los aspectos físicos de un sistema.
- Modela la vista de implementación estática de un sistema.
- Modela los elementos físicos que residen en un nodo, tales como ejecutables, tablas, librerías, archivos y documentos.
- Un Diagrama de Componentes muestra un conjunto de componentes y sus relaciones.

Los elementos que lo componen son:

- Componentes
- Interfaces
- Relaciones de dependencia, generalización, asociación, realización.

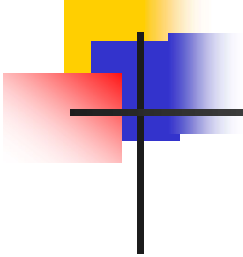
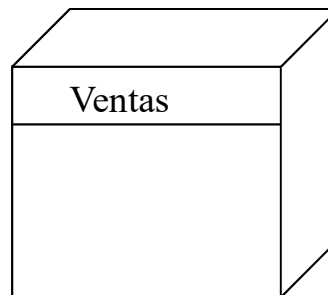


Diagrama de Despliegue

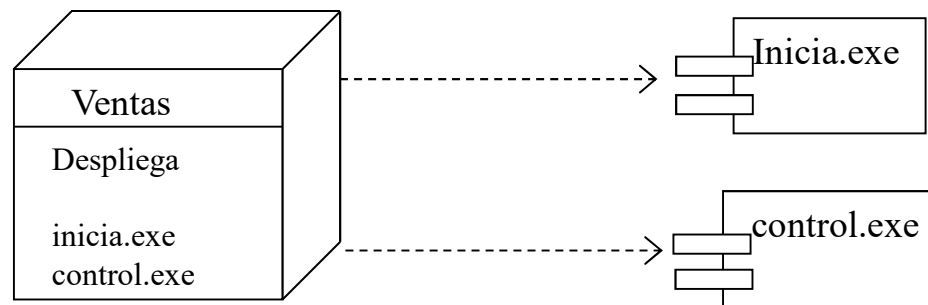
UML – NODO

- Es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene alguna memoria y capacidad de procesamiento.
- Posee un nombre simple, *ej: Ventas* o un nombre extendido indicando el paquete que lo contiene, *ej: servidor::Ventas*.
- Gráficamente:



UML – NODO

- En los Nodos se ejecutan los Componentes.
- La relación entre un nodo y un componente se puede modelar con una relación de dependencia.
- Los nodos se pueden organizar agrupándolos en paquetes. También a través de relaciones de dependencia, generalización, asociación, agregación. Generalmente se conectan con una asociación.





UML – Diagrama de Despliegue

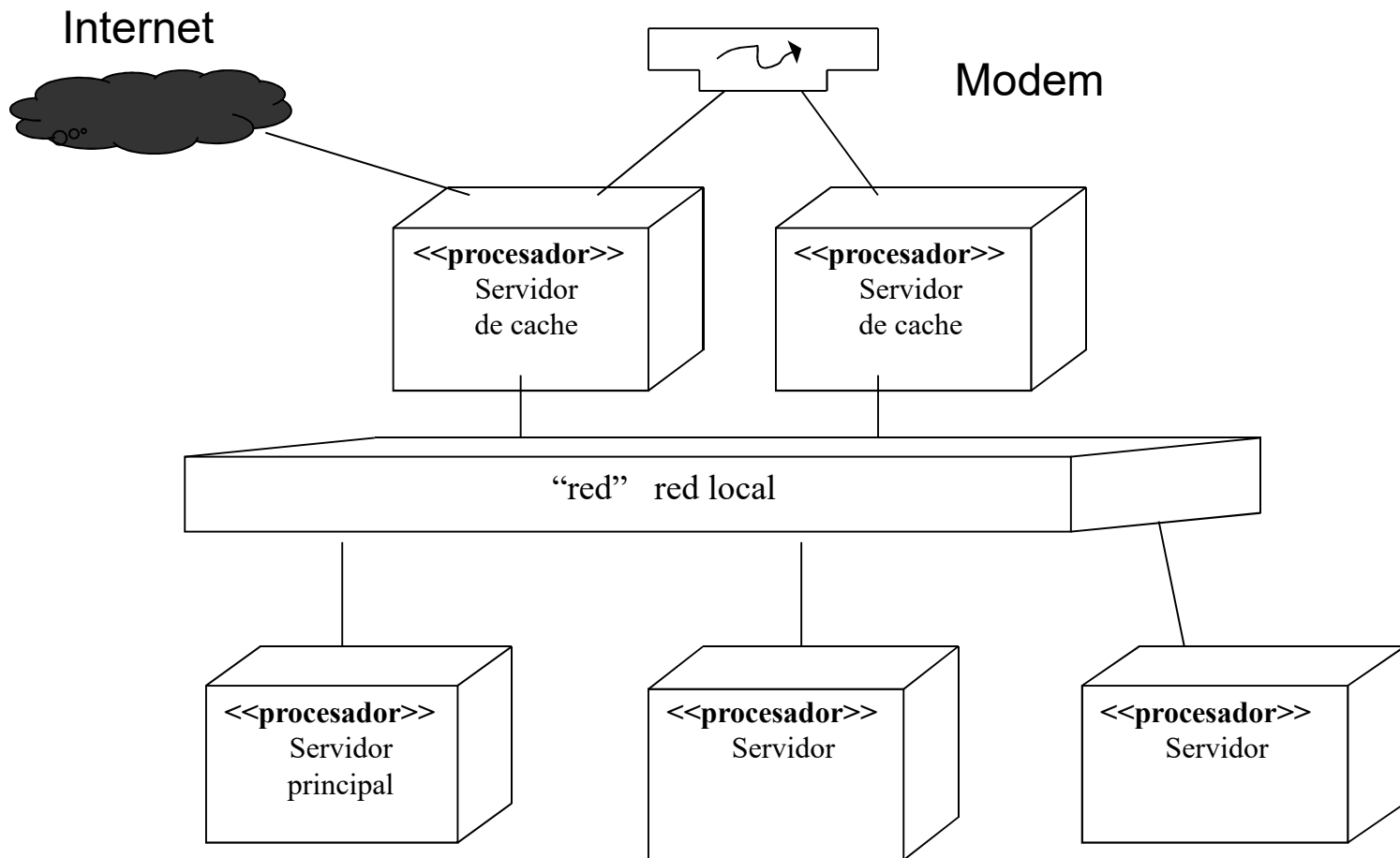
- Modela aspectos físicos de un sistema.
- Modela la vista de despliegue estática de un sistema.
- Modela una configuración de nodos y los componentes que residen en ellos.
- Modela la topología del hardware donde se ejecuta el sistema.

Los elementos que lo componen son:

- Nodos
 - Relaciones de dependencia, generalización, asociación y realización.
 - Pueden contener los componentes que residen en los nodos.
- Nota: UML no es un lenguaje de descripción de hardware de propósito general como VHDL.

UML – Diagrama de Despliegue

➤ Ejemplo:



Bibliografía Consultada



- Capítulos 26, 29 y 30: Booch Grady, Rumbaugh James, Jacobson Ivar. El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. 2006.
- <http://www.uml.org>

