Modelos de Dominio

Fundamentos de Ingeniería de Software/Análisis y Diseño de Software

Pablo Cruz Navea – Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María

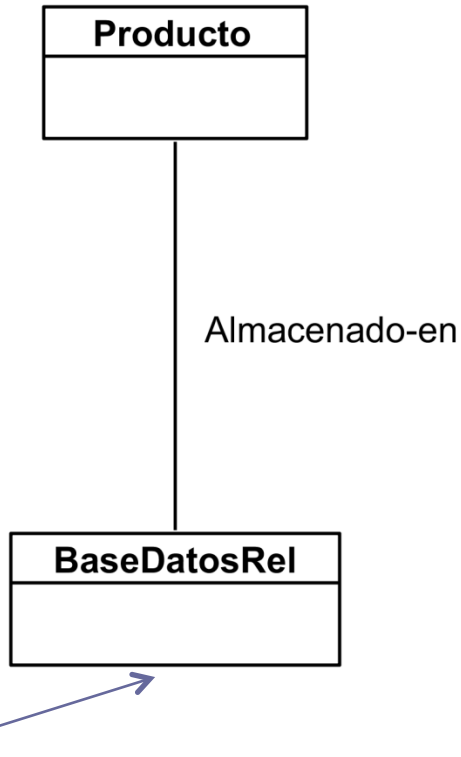
Motivación

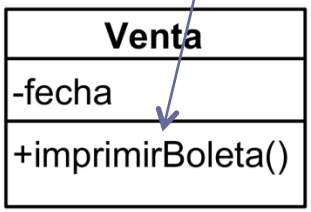
- La escritura de **casos de uso** y **requerimientos** es sencilla cuando nos enfrentamos a dominios de trabajo conocidos
  - ¿Qué hacemos cuando entramos a dominios no tan conocidos?
- Incluso cuando estamos ante dominios conocidos, la existencia de una cultura organizacional distinta en cada dominio fomenta la existencia de distintos conceptos para (muchas veces) similares significados

Qué es un Modelo de Dominio

- Diagrama que ilustra **clases (objetos) significativas** de un **dominio de problema** particular para el analista y diseñador
  - Idea clave: clases (objetos) del mundo real, **NO** clases o componentes del software
- Objetivo: visualizar y comprender la relación de los conceptos que se manejan en el dominio del problema
  - Es esencial mantener un adecuado nivel de abstracción, es decir, ignorar lo que no es interesante en el dominio
- UML ofrece notación de diagrama de clases para modelos de dominios
  - A diferencia de los modelos de clases de software (que veremos más adelante), los modelos de dominio no incluyen operaciones (métodos)

Incorrecto: clases no llevan métodos/operaciones





Incorrecto: clases no son componentes o elementos del software

Caracterizando los Modelos de Dominio

- Clases conceptuales (representan objetos del dominio)
  - Personas, animales, cosas, organizaciones…
  - Ej: cajero, encuestador, cliente, factura, bodega
- Asociaciones entre los objetos y/o clases
  - Ej: cliente **firma** factura
- Multiplicidad en las asociaciones
  - Ej: un cliente **firma** muchas facturas
- Atributos de las clases conceptuales
  - Ej: cliente **(RUT, nombre, apellido, dirección)**

Ejemplos de clases conceptuales

Atributos [1]

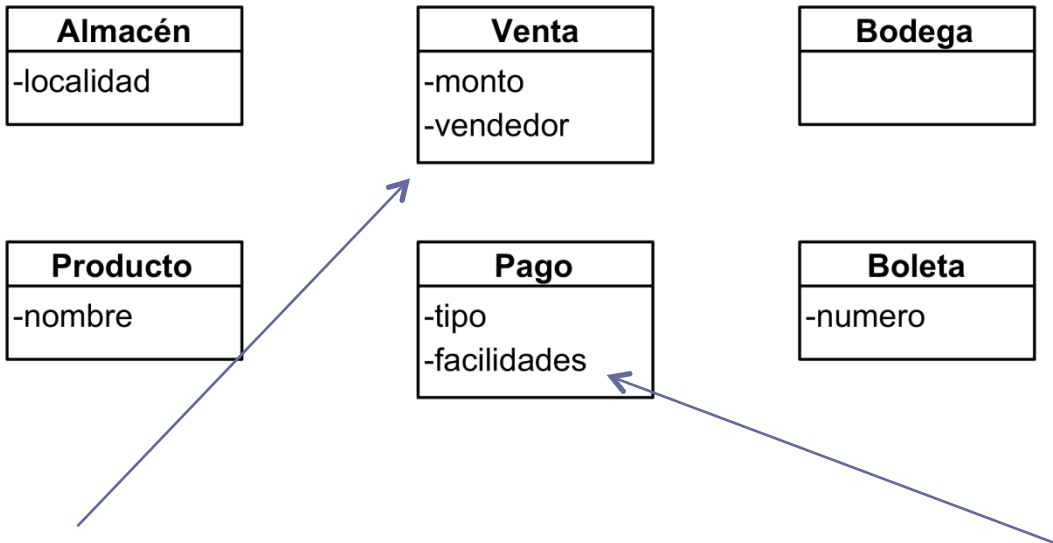
- En el contexto de modelos de dominio, un atributo es un valor (texto, número, etc.) que caracteriza a una clase
- Las clases pueden tener muchos atributos, pero no deben incluirse todos los posibles
  - Los casos de uso son una buena fuente de información para decidir qué atributos deben ser incluidos
    - Se incluirán aquellos que sea necesario recordar para el contexto

Atributos [2]

- En UML, los atributos van en el segundo compartimento de una clase (caja)
- Se sugiere que los atributos sean tipos de datos simples
  - Si un atributo no es un tipo de dato simple, probablemente estemos frente a una clase
  - Esto no implica que los atributos en las clases del software deban ser tipos de datos primitivos
- Si un atributo no es un dato simple corremos el riesgo de relacionar clases con atributos
  - Por ejemplo, es mejor relacionar dos clases **Vuelo** Viaja-a **Destino** que tener una sola clase **Vuelo (destino)**
    - Pero siempre depende del contexto o dominio
- Por lo tanto, las clases se relacionan con **asociaciones** y no atributos
  - Tampoco se puede usar **atributos foráneos**
    - Un modelo de dominio **no** es un modelo de datos relacional

Ejemplos de atributos

Una clase (concepto) puede no tener atributos



Dependerá de la situación particular si un "vendedor" debe considerarse como atributo o clase conceptual

Facilidades: nombre genérico para "número de cuotas"

Error común: clase conceptual aparece como atributo

- Un error típico (a evitar) es confundir una clase conceptual con un atributo
- **Regla general:** si un elemento no es entendido como número o texto, ese elemento es ***probablemente*** una clase (y no atributo)
  - Ej: Teléfono (número → atributo) versus Teléfono (dispositivo → clase)
- Si las dudas persisten, es preferible modelar el elemento como clase
  - Recordar que el modelo de dominio es un modelo de conceptos (clases) por lo que los atributos debieran ser escasos (en comparación a la cantidad de clases)
    - Esto implica que debemos dudar de un modelo lleno de atributos y pocas clases

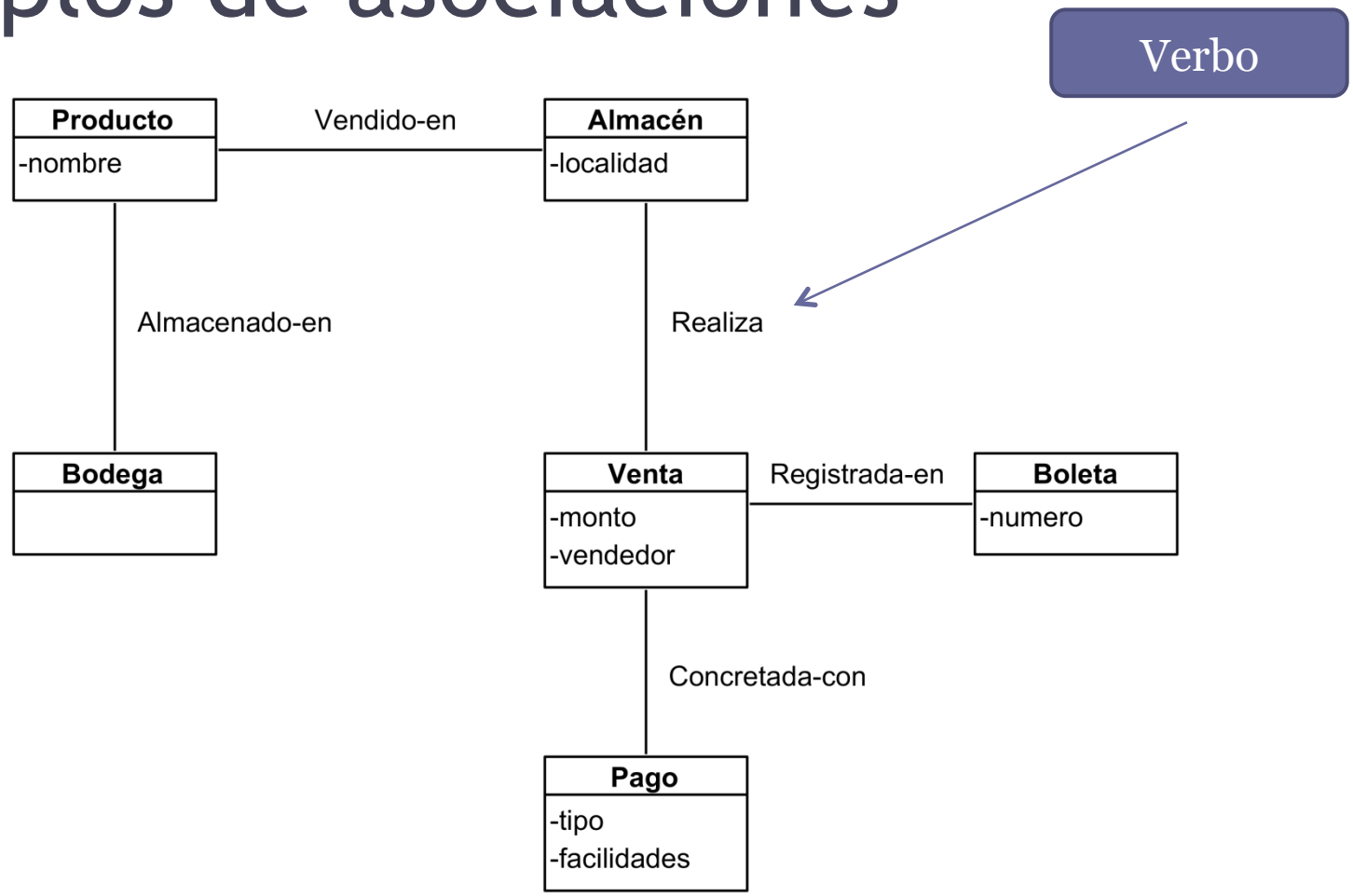
Asociaciones [1]

- Una asociación es una relación entre los conceptos del modelo de dominio
- Deben indicar alguna conexión significativa, que tenga sentido recordarla
  - Es buena idea preguntarse: *¿vale la pena recordar esta relación en el tiempo?*
  - Por ejemplo, si la relación "Producto **Almacenado-en** Bodega" ya existe, no tiene sentido anotar otra relación que diga "Bodega **Almacena** Producto"
    - El lenguaje UML permite esto, pero no tiene utilidad
- Las asociaciones se representan en UML por una línea que une dos o más clases y que es inherentemente bidireccional
  - Pero esta conexión bidireccional debe entenderse de manera abstracta puesto que no implica conexión alguna de componentes de software

Asociaciones [2]

- Las asociaciones cuentan con un nombre escrito (un verbo que indica acción)
  - Por ejemplo:
    - Pagado-por, Enviado-a, Almacenado-en, Firma-en
    - PagadoPor, EnviadoA, AlmacenadoEn, FirmaEn
- Por convención, la lectura del nombre de la asociación es de izquierda a derecha y de arriba hacia abajo
  - Sin embargo, UML no especifica reglas para esto
  - Es simplemente una ayuda para la lectura del modelo

Ejemplos de asociaciones



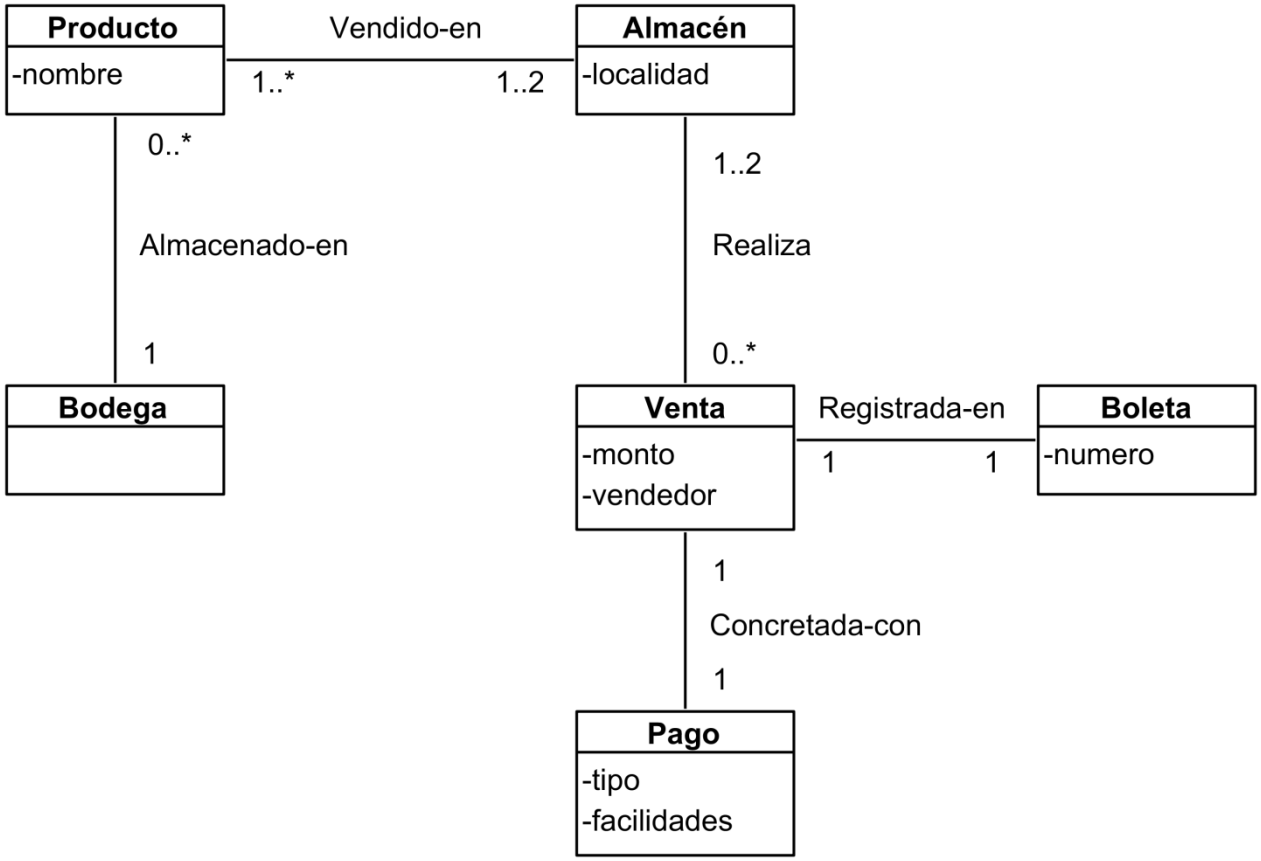
Asociaciones: multiplicidad [1]

- Cada extremo de una línea de asociación se denomina **Rol**
- Y cada **Rol** tiene asociada una **multiplicidad**
  - Define **cuántas instancias** de una clase A pueden ser asociadas con una instancia de una clase B
- **Importante:** por convención, la multiplicidad se define en un momento particular del tiempo (dependiendo del contexto) y no sobre espacios de tiempo arbitrarios
  - Por ejemplo, **1** hombre **se casa** con **1** mujer
    - Así se concibe legalmente el matrimonio (expresado con la relación "se casa"), independiente de que en un espacio de tiempo arbitrario un hombre pueda casarse con más mujeres (o viceversa)

Asociaciones: multiplicidad [2]

- El analista debe ser cuidadoso con la multiplicidad puesto que puede influir en decisiones importantes al momento de construir el software
- La multiplicidad se expresa por números y/o comodines, por ejemplo:
  - * : cero o más (muchas)
  - **1..*** : una o más
  - **1..40** : una a cuarenta
  - **5** : exactamente 5
  - **3, 5, 8** : exactamente 3, 5, u 8

Ejemplos de multiplicidad



Comentarios sobre el modelo [1]

- El ejemplo muestra un modelo que puede ser completo o parte de otro modelo más grande
- El modelo nos dice, entre otras cosas, que:
  - En un momento particular de tiempo, *cero o más* instancias de **Producto** (por ejemplo: P1, P2, P3) pueden estar almacenados en *una* instancia de **Bodega** (por ejemplo: B1)
    - O que *solo una* Bodega puede tener cero o más instancias de Producto
  - Vendedor es un número (para el negocio de ejemplo) y no una clase
    - Es compatible con la idea de que, para el dominio, la venta es realizada por el **Almacén**
    - Si esto no es cierto, ¡el analista debe revisar el modelo y considerar vendedor como una clase!

Comentarios sobre el modelo [2]

- También nos dice que…
  - Para el contexto, la venta es "realizada" en una boleta
    - Esto es, se considera una venta (que puede contener varios productos) como una sola y, por lo tanto, una boleta
  - La venta "existe" sólo cuando se realiza el pago, independiente del número de cuotas (facilidades de pago)
- Ejercicio interesante: ¿qué más le dice a usted este modelo? ¿está de acuerdo con *todo* lo que describe este modelo?

En resumen…

- Un modelo de dominio es una representación gráfica de los conceptos, sus atributos y sus relaciones en el dominio del problema
- No existen modelos absolutamente correctos o incorrectos
  - Pero sí existen modelos más útiles que otros
- Asociaciones y atributos son elementos secundarios
  - El esfuerzo debe estar en encontrar clases, conceptos
  - Así también, las asociaciones y atributos deben agregarse en la medida que aporten al entendimiento del dominio
    - Cuidado: ¡Es muy fácil caer en la tentación de llenar el modelo con asociaciones y atributos con el fin de que parezca “mejor” que otros!

FIN