MODELO DEL DOMINIO

Extraído de:
UML y Patrones. 2ª Edición.
Craig Larman.
Prentice Hall. 2003

1. Introducción

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software, y será una entrada necesaria para varios de los siguientes artefactos que se verán en este curso. El modelo del dominio muestra (a los modeladores) clases conceptuales significativas en un dominio de] problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Este documento presenta técnicas introductorias para la creación de modelos del dominio.

IMPORTANTE: Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades.

La etapa orientada a objetos esencial del análisis o investigación es la descomposición de un dominio de interés en clases conceptuales individuales u objetos -las cosas de las que, somos conscientes-. Un **modelo del dominio** es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina **modelos conceptuales** (término utilizado en la primera edición del libro de Larman), **modelo de objetos del dominio** y **modelos de objetos de análisis**.

Utilizando la notación UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Pueden mostrar:

- Objetos del dominio o clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de las clases conceptuales.

Por ejemplo, la Figura 1 muestra un modelo del dominio parcial. Ilustra que las clases conceptuales *Pago* y *Venta* son significativas en este dominio, que un *Pago* está relacionado con una *Venta* de un modo que es significativo hacer notar, y que una Venta tiene una fecha y una hora.

La Figura 1 visualiza y relaciona algunas palabras o clases conceptuales del dominio. También describe una abstracción de las clases conceptuales, porque hay muchas cosas que uno podría comunicar sobre los registros, las ventas, etc. El modelo muestra una vista parcial o abstracción, e ignora detalles sin interés (para el modelador).

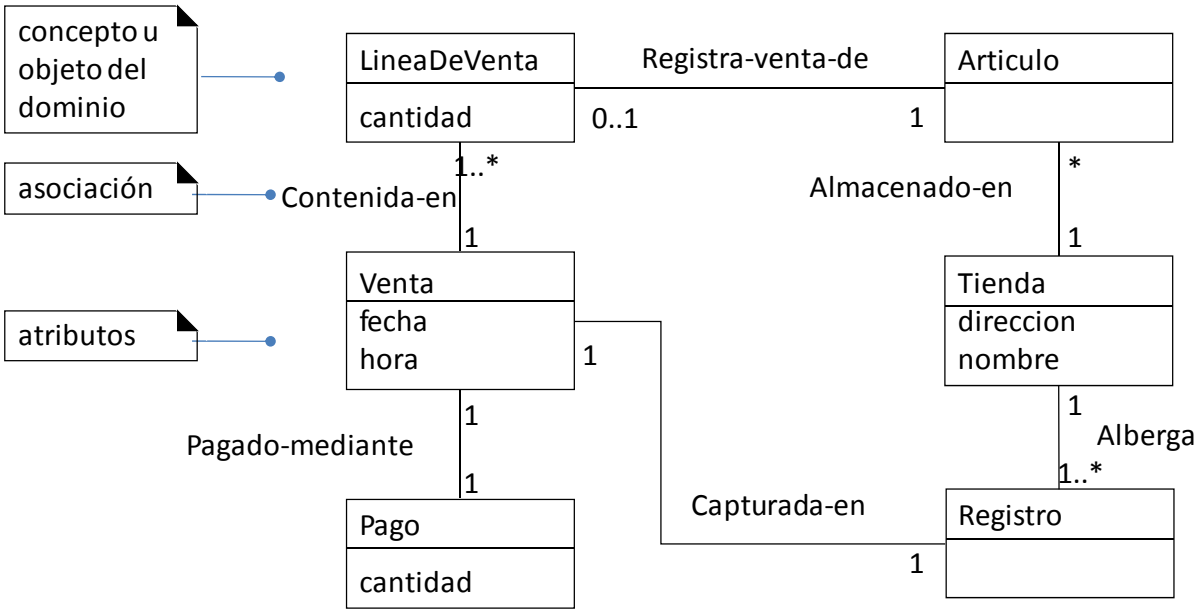

Figura 1. Modelo del dominio parcial-un diccionario visual-.

La información que presenta (utilizando la notación UML) podría, de manera alternativa, haberse expresado en prosa, mediante sentencias. Pero es fácil entender los distintos elementos y sus relaciones mediante este lenguaje visual, puesto que un porcentaje significativo del cerebro toma parte en el procesamiento visual --es una cualidad de los humanos-.

Por tanto, el modelo del dominio, como muestra la Figura 2, podría considerarse como un *diccionario visual* de las abstracciones relevantes, vocabulario del dominio e información del dominio. Un modelo del dominio es una representación de las cosas del mundo real del dominio de interés, no de componentes software, como una clase Java o C+ + (ver Figura 2), u objetos software con responsabilidades. Por tanto, los siguientes elementos no son adecuados en un modelo del dominio:

- Artefactos software, como una ventana o una base de datos, a menos que el dominio que se esté modelando sea de conceptos software, como un modelo de interfaces de usuario gráficas. I

- Responsabilidades o métodos.

2. Clases Conceptuales

El modelo del dominio muestra las clases conceptuales o vocabulario del dominio. In formalmente, una clase conceptual es una idea, cosa u objeto. Más formalmente, una clase conceptual podría considerarse en términos de su símbolo, intensión, y extensión:

- **Símbolo**: Palabras o imágenes que representan una clase conceptual.

- **Intensión:** La definición de una clase conceptual.

- **Extensión**: El conjunto de ejemplos a los que se aplica la clase conceptual.

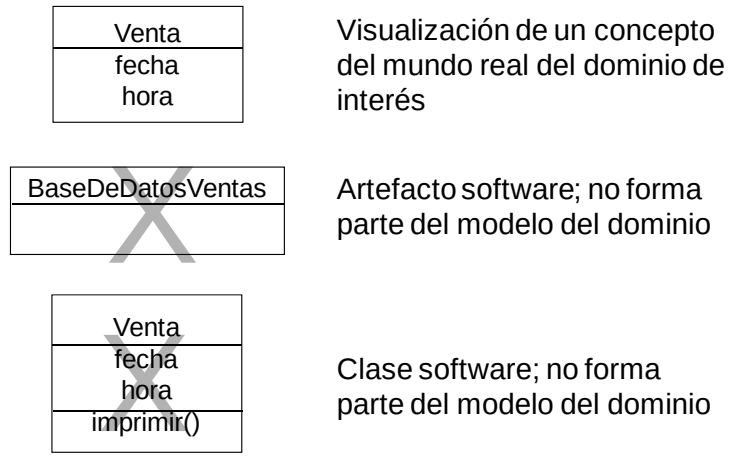

Figura 2. Modelo del dominio parcial-un diccionario visual-.

Por ejemplo, considere la clase conceptual para el evento de una transacción de compra. Se podría elegir nombrarla con el símbolo *Venta*. La intensión de una Venta podría establecer que "representa el hecho de una transacción de una compra, y tiene una fecha y una hora". La extensión de una Venta la forman todos los ejemplos de ventas; en otras palabras, el conjunto de todas las ventas.

Cuando creamos un modelo del dominio, normalmente, el símbolo y la vista intensional de la clase conceptual son los que tienen un mayor interés práctico.

Los problemas del software pueden ser complejos; la descomposición --divide y vencerás- es una estrategia común para tratar esta complejidad mediante la división del espacio del problema en unidades fáciles de comprender. En el análisis estructurado, la dimensión de la descomposición es por procesos o por funciones. Sin embargo, en el análisis orientado a objetos, la dimensión de la descomposición es fundamentalmente por cosas o entidades del dominio. Una diferencia esencial entre el análisis orientado a objetos y el estructurado es: la división por clases conceptuales (objetos) en lugar de la división por funciones. Por tanto, la principal tarea del análisis es identificar diferentes conceptos en el dominio del problema y documentar el resultado en un modelo del dominio.

Por ejemplo, en el dominio de ventas en una tienda del mundo real, existen las clases conceptuales de *Tienda*, *Registro* y *Venta*. Por tanto, nuestro modelo del dominio, podría incluir *Tienda*, *Registro* y *Venta*.

2.1. Identificación de las clases conceptuales

Nuestro objetivo es crear un modelo del dominio de clases conceptuales interesantes significativas del dominio de interés (ventas). En este caso, eso significa conceptos relacionados con el caso de uso Procesar Venta.

En el desarrollo iterativo, uno incrementalmente construye un modelo del dominio lo largo de varias iteraciones en la fase de elaboración. En cada una, el modelo de dominio se limita a los escenarios anteriores y actual en estudio, en lugar de un modelo de "gran explosión", que en las primeras etapas intenta capturar todas las posibles clases conceptuales y las relaciones. Por

ejemplo, esta iteración está limitada al escenario (pago en efectivo de Procesar Venta; por tanto, se creará un modelo de dominio parcial; únicamente para reflejar eso -no más-. La tarea central es, por tanto, identificar las clases conceptuales relacionadas con escenario que se está diseñando.

A continuación se presenta una guía útil para la identificación de las clases conceptuales: Es mejor especificar en exceso un modelo del dominio con muchas clases conceptuales de grano fino que especificar por defecto. No se debe pensar que un modelo de dominio es mejor si contiene pocas clases conceptuales suele ser verdad justamente lo contrario.

Es normal obviar clases conceptuales durante la etapa de identificación inicial, y descubrirlas más tarde al considerar los atributos y asociaciones, o durante el trabajo de diseño. Cuando se encuentren, se pueden añadir al modelo del dominio.

No conviene excluir una clase conceptual simplemente porque los requisitos no indican una necesidad obvia para registrar información sobre ella (un criterio común en el mi dejado de datos para el diseño de bases de datos relacionales, pero no relevante en el modelado del dominio), o porque la clase conceptual no tiene atributos. Asimismo, es válido tener clases conceptuales sin atributos, o clases conceptuales con un rol meramente de comportamiento en el dominio, en lugar de un rol de información.

A continuación se presentan dos técnicas para la identificación de las clases conceptuales:

1. Utilización de una lista de categorías de clases conceptuales.
2. Identificación de frases nominales.

Otra técnica para el modelado del dominio (que no se verá aquí) es el uso de patrones de análisis, que son modelos de dominios parciales existentes creados por expertos, utilizando libros publicados sobre el tema.

2.1.1. Utilización de Una Lista de Categorías de Clases Conceptuales

Se comienza la creación de un modelo del dominio haciendo una lista de clases conceptuales candidatas. La Tabla l contiene categorías habituales que, normalmente, merece la pena tener en cuenta, aunque no en ningún orden particular de importancia. Los ejemplos se han extraído del dominio de las tiendas y las reservas de vuelos.

Tabla 1. Lista de categorías de clases conceptuales.

Categoría de cíase conceptual	*Ejemplos*
objetos tangibles o físicos	*Registro* *Avion*
especificaciones, diseños o descripciones de la cosas	*EspecifiacionDelProducto* *DescripcionDelVuelo*
lugares	*Tienda*
transacciones	*Venta, Pago* *Reserva*
líneas de la transacción	*LíneaDeVenta*
roles de la gente	*Cajero* *Píloto*

contenedores de otras cosas	*Tienda, Lata* *Avion*
cosas en un contenedor	*Articulo* *Pasajero*
otros sistemas informáticos o electromecánicos externos al sistema	*SistemaAutorizadonPagoCredíto* *ControlDeTraficoAereo*
conceptos abstractos	*Ansía* *Acrofobía*
organizaciones	*DepartamentoDeVentas* *CompañiaAerea*
hechos	*Venta,Pago,Reunion* *Vuelo, Colision, Aterrízaje*
procesos (normalmente *no* se representan como conceptos, pero podría ocurrir)	*VentaDeUnProducto* *ReservaUnAsiento*
reglas y políticas	*PoliticaDeReintegro* *PoliticaDeCance/ación*
catálogos	*CatalogoDeProductos* *CatalogoDePiezas*
registros de finanzas, trabajo, contratos, cuestiones legales	*Recibo, LibroMayor, ContratoEmpleo* *RegistroMantenimiento*
instrumentos y servicios financieros	*LineaDeCredito* *Stock*
manuales, documentos, artículos de referencia, libros	*ListaDeCambiosDePreciosDiarios* *Manua/Reparaciones*

2.1.2. Identificación de Frases Nominales

Otra técnica útil (debido a su simplicidad) es el análisis lingüístico: identificar los nombres y frases nominales en las descripciones textuales del dominio, y considerarlos como clases conceptuales o atributos candidatos.

Se debe tener cuidado con este método; no es posible realizar una correspondencia mecánica de nombres a clases, y las palabras en lenguaje natural son ambiguas.

En cualquier caso, es otra fuente de inspiración. Los casos de uso en formato completo constituyen una descripción excelente a partir de la cual extraer este análisis. Por ejemplo, se puede utilizar el escenario actual del caso de uso Procesar Venta.

Escenario principal de éxito (o Flujo Básico):

1. El **Cliente** llega a un **terminal PDV** con **mercancías** y/o **servicios** que comprar.

2. El **Cajero** comienza una nueva **venta**.

3. El **Cajero** introduce el **identificador del artículo**.

4. El Sistema registra la **línea de la venta** y presenta la **descripción del artículo**, **precio** y **suma** parcial. El precio se calcula a partir de un conjunto de reglas de precios.

El Cajero repite los pasos 3-4 hasta que se indique.

5. El Sistema presenta el total con los **impuestos** calculados.

6. El Cajero le dice al Cliente el total y solicita el **pago**.

7. El Cliente paga y el Sistema gestiona el pago.

8. El Sistema registra la **venta** completa y envía la información de la venta y el pago sistema de **Contabilidad** externo (para la contabilidad y las **comisiones**) y al sistema de **Inventario** (para actualizar el inventario).

9. El Sistema presenta el **recibo**.

10. El Cliente se va con el recibo y las mercancías (si es el caso).

Extensiones (o Flujos Alternativos):

7a. Pago en efectivo:

1. El Cajero introduce la **cantidad** de dinero **entregada** en efectivo.

2. El Sistema muestra la **cantidad** de dinero a **devolver** y abre el **cajón de caja**.

3. El Cajero deposita el dinero entregado y devuelve el cambio al Cliente.

4. El Sistema registra el pago en efectivo.

"El modelo del dominio es una visualización de los conceptos del dominio y vocabulario relevantes. ¿Dónde se encuentran estos términos? En los casos de uso. Por tanto, constituyen una fuente rica a explorar mediante la identificación de frases nominales.

Algunas de las frases nominales son clases conceptuales candidatas, algunas podrían hacer referencia a clases conceptuales que se ignoran en esta iteración (por ejemplo, "Contabilidad" y "comisiones"), y algunas podrían ser atributos de las clases conceptuales. Más adelante se presentarán algunos consejos que permiten diferenciar entre los dos.

Un punto débil de este enfoque es la imprecisión del lenguaje natural; frases nominales diferentes podrían representar la misma clase conceptual o atributo, entre otras ambigüedades. Sin embargo, se recomienda que se combine con la técnica la Lista de Categorías de Clases Conceptuales.

2.2. Clases Conceptuales Candidatas para el Dominio de Ventas

A partir del análisis de la Lista de Categorías de Clases Conceptuales y las frases nominales, se genera una lista de clases conceptuales candidatas del dominio. La lista está restringida a los requisitos y simplificaciones que se están estudiando actualmente -el escenario simp1ificado de *Procesar Venta*-.

Registro	EspecificacionDelProducto
Articulo	LineaDeVenta

Tienda	Cajero
Venta	Cliente
Pago	Encargado
CatalogoDeProductos	

No existe una lista "correcta". Es una colección algo arbitraria de abstracciones y vocabulario del dominio que el modelador considera relevantes. En cualquier caso, siguiendo la estrategia de identificación, diferentes modeladores producirán listas similares.

Un recibo es un informe de una venta y del pago, y una clase conceptual relativamente destacable del dominio, por tanto, ¿debería mostrarse en el modelo? A continuación se presentan algunos factores a tener en cuenta:

- Un recibo es un informe de una venta. En general, no es útil mostrar un informe de otra información en un modelo del dominio puesto que toda esta información se deriva de otras fuentes; duplica información que se encuentra en otras partes. Esta es una razón para excluido.

- Un recibo tiene un rol especial en término de las reglas del negocio: normalmente, confiere al portador del recibo, el derecho a devolver los artículos comprados. Esta es una razón para mostrado en el modelo.

Puesto que la devolución de artículos no está siendo considerada en esta iteración, el *Recibo* será excluido. Durante la iteración que aborda el caso de uso Gestionar Devoluciones, estaría justificada su inclusión.

2.3. Clases Conceptuales de Especificación o Descripción

La siguiente discusión podría, al principio, parecer relacionada con un tema raro y altamente especializado. Sin embargo, prueba que la necesidad de las clases conceptuales de especificación (como se definirán) es habitual en muchos modelos del dominio. Por tanto, se destaca.

Dado lo siguiente:

- Una instancia de un *Articulo* representa un objeto físico de una tienda; como tal, podría incluso tener un número de serie.

- Un *Articulo* tiene una descripción, precio, e identificador del artículo (articuloID), que no se recogen en ningún otro sitio

- Todo el mundo que trabaja en la tienda tiene amnesia.

- Cada vez que se vende un artículo físico real, se elimina una instancia de *Articulo* software correspondiente, del "terreno del software".

Con estas suposiciones, ¿qué ocurre en el siguiente escenario?

Existe una fuerte demanda de las nuevas hamburguesas vegetales que han tenido mucho éxito

-ObjetoHamburguesa-. La tienda las vende todas, lo que implica que se eliminen de la memoria del ordenador todas las instancias de *Articulo* de los ObjetoHamburguesa.

Ahora, aquí está lo esencial del problema: Si alguien pregunta: "¿Cuánto cuestan los ObjetosHamburguesa?", nadie puede contestar, porque el precio se encontraba en las instancias inventariadas, que se eliminaron cuando se vendieron.

Nótese también que el modelo actual, si se implementa en el software tal y como se describe, contiene datos duplicados y es ineficiente en cuanto al espacio, puesto que la descripción, el precio y el ID, se duplican en cada instancia de Articulo del mismo producto.

El problema anterior ilustra la necesidad de conceptos de objetos que sean especificaciones o descripciones de otras cosas. Para solucionar el problema del *Articulo*, lo que se necesita es una clase conceptual EspecificacionDelProducto (o EspecificacionDelArticulo, DescripcionDelProducto...,) que recoge la información sobre los artículos. Una EspecificacionDelProducto no representa un *Articulo*, sino una descripción de la información sobre los artículos. Nótese que incluso si todos los elementos inventariados se venden, y sus correspondientes instancias software de Articulo se eliminan, permanece todavía la EspecificacionDelProducto.

Los objetos de descripción o especificación están fuertemente relacionados con las cosas que describen. En un modelo del dominio, es típico establecer que una EspecificacionDeX Describe un X (ver Figura 3).

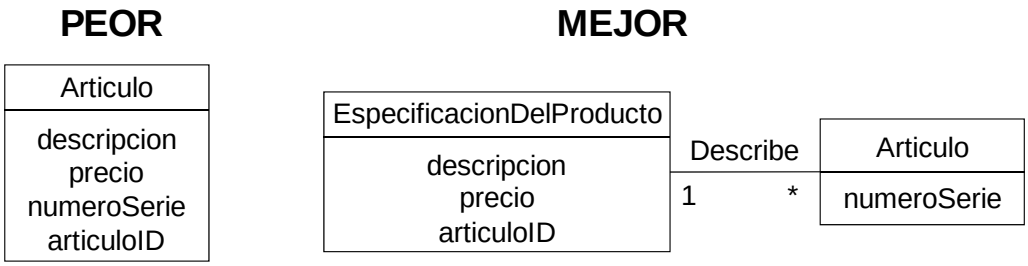

Figura 3. Especificaciones o descripciones sobre otras cosas.

La necesidad de clases conceptuales de especificación es habitual en los dominios de ventas y productos. También es típica en la fabricación, donde se requiere una descripción de la cosa que se fabrica, que es distinto de la cosa en sí misma, Se ha dedicado tiempo y espacio a motivar las clases conceptuales de especificación porque son muy frecuentes; no es un concepto de modelado raro.

A continuación se sugiere sugieren cuándo utilizar las especificaciones. Conviene añadir una clase conceptual de especificación o descripción (por ejemplo, EspecificacionDelProducto) cuando:

- Se necesita la descripción de un articulo o servicio, independiente de la existencia actual de algún ejemplo de esos articulas o servicios.

- La eliminación de instancias de las cosas que describen (por ejemplo, Articulo) dan como resultado una pérdida de información que necesita mantenerse, debido a la asociación incorrecta de información con la cosa eliminada.

- Reduce información redundante o duplicada.

2.4. / Ejemplo: el Modelo del Dominio del PDV NuevaEra

La lista de clases conceptuales generadas para el dominio del PDV NuevaEra se podría representar gráficamente (ver Figura 4) para mostrar el comienzo del Modelo del Dominio.

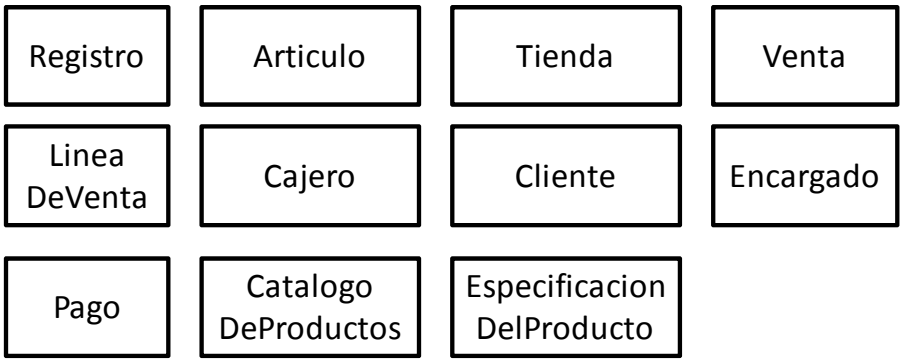

Figura 4. Modelo del dominio inicial.

Las consideraciones acerca de los atributos y asociaciones del Modelo del Dominio_ se posponen a secciones posteriores.

3. Asociaciones

Una asociación es una relación entre tipos (o más concretamente, instancias de estos tipos) que indica alguna conexión significativa e interesante. En UML, las asociaciones se definen como "la relación semántica entre dos o más clasificadores que implica conexiones entre sus instancias".

Resulta útil identificar aquellas asociaciones entre clases conceptuales que son necesarias para satisfacer los requisitos de información de los escenarios actuales que se están desarrollando, y que ayudan a entender el modelo del dominio. Esta sección explora la identificación de las asociaciones adecuadas, y añade las asociaciones al modelo del dominio del caso de estudio NuevaEra.

Las asociaciones que merece la pena registrar, normalmente, implican conocimiento de una relación que es necesario conservar durante algún tiempo -podrían ser milisegnndos o años, dependiendo del contexto-. En otras palabras, ¿entre qué objetos necesitamos mantener en memoria una relación? Por ejemplo, ¿necesitamos recordar qué instancias de una *LineaDeVenta* están asociadas con una instancia de una *Venta*? Sin ninguna duda, de otra manera no sería posible reconstruir una venta, imprimir un recibo o calcular el total del! venta.

Considere la inclusión de las siguientes asociaciones en un modelo del dominio:

- Asociaciones de las que es necesario conservar el conocimiento de la relación durante algún tiempo (asociaciones "necesito-conocer").

- Asociaciones derivadas de la Lista de Asociaciones Comunes.

Por el contrario, ¿necesitamos recordar una relación entre la *Venta* actual y un Encargado? No, los requisitos no dan a entender que se necesite ninguna relación de este tipo. No es incorrecto mostrar una relación entre una *Venta* y un *Encargado*, pero no es convincente o útil en el contexto de nuestros requisitos.

Éste es un punto importante. En un modelo del dominio con n clases del dominio diferentes, pueden existir n(n-1) asociaciones entre diferentes clases conceptuales –un número potencialmente grande-. Muchas líneas en un diagrama añadirán "ruido visual" y lo hará menos comprensible. Por tanto, se debe ser cuidadoso al añadir líneas de asociación. Se recomienda utilizar como criterio las guías que se sugieren en esta sección.

3.1. Notación de las Asociaciones en UML

Una asociación se representa como una línea entre clases con un nombre de asociación, La asociación es inherentemente bidireccional, lo que significa que desde las instancias de cualquiera de las dos clases, es posible el recorrido lógico hacia la otra. Este recorrido es puramente abstracto; no se trata de una sentencia sobre conexiones entre entidades software.
.

Los extremos de la asociación podrían contener una expresión de multiplicidad que indica la relación numérica entre las instancias de las clases.

Una "flecha de dirección de lectura" opcional indica la dirección de la lectura del nombre de la

asociación; no indica la dirección de la visibilidad o navegación. Si no está presente, la convención es leer la asociación de izquierda a derecha o de arriba hacia abajo, aunque no es una regla de UML.

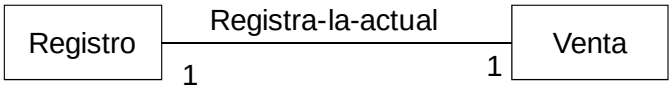

Figura 5. Notación UML para las asociaciones.

La flecha de dirección de lectura no tiene significado en términos del modelo; sólo es una ayuda para el lector del diagrama.

3.2. Localización de las Asociaciones: Lista de Asociaciones Comunes

Se puede comenzar la inclusión de asociaciones utilizando la lista de la Tabla 2. Contiene categorías comunes que, normalmente, merece la pena tener en cuenta. Los ejemplos se han extraído de los dominios de ventas y reservas de vuelos.

Tabla 2. Lista de asociaciones comunes.

Categoría	*Ejemplos*
A es una parte física de B	Cajon-Reglstro (o más concretamente, TPDV) Ala-Avion
A es una parte lógica de B	LineaDeVenta- Venta EtapaVuelo-RutaVuelo
A está contenido físicamente en B	Registo-Tienda, Articulo-Estanteria Pasajero-Avion
A está contenido lógicamente en B	DescripcionDelArticulo-Catalogo Vuelo-PlaniticacionVuelo
A es una descripción de B	*DescripcionDelArticulo-Articulo* *DescripcionDelVuelo- Vuelo*
A es un línea de una transacción o informe de B	*LineaDeVenta-Venta* *TrabajoMantenimiento-RegistroDe- Mantenimiento*
A se conoce/registra/recoge/informa/ captura en B	*Venta-Registro* *Reserva-ListaPasajeros*
A es miembro de B	*Cajero- Tienda* *Piloto-CompañiaAerea*
A es una subunidad organizativa de B	*Departamento- Tienda* *Mantenimiento-CompañiaAerea*
A utiliza o gestiona B	*Cajero-Registro* *Piloto-Avion*
A se comunica con B	*Cliente-Cajero* *AgenteDeReservas-Pasajero*
A está relacionado con una transacción B	*Cliente-Pago* *Pasajero-Billete*
A es una transacción relacionada con otra transacción B	*Pago-Venta* *Reserva-Cancelacion*
A está aliado de B	*LineaDe Venta-LineaDe Venta*

	Ciudad-Ciudad
A es propiedad de B	*Registro- Tienda* *Avion-CompañiaAerea*
A es un evento relacionado con B	*Venta-Cliente, Venta- Tienda* *Salida-Vuelo*

Algunas de las categorías de asociaciones de prioridad alta que son, invariablemente, útiles incluirlas en un modelo del dominio:

- A es una parte lógica o física de B.
- A está contenida física o lógicamente en B.
- A se registra en B.

3.3. Guías para las Asociaciones

- Es conveniente centrarse en aquellas asociaciones para las que se necesita conservar el conocimiento de la relación durante algún tiempo (asociaciones "necesito-conocer").
- Es más importante identificar clases conceptuales que identificar asociaciones.
- Demasiadas asociaciones tienden a confundir un modelo del dominio en lugar de aclararlo. Su descubrimiento puede llevar tiempo, con beneficio marginal.
- Se debe evitar mostrar asociaciones redundantes o derivadas.

3.4. Roles

Cada extremo de una asociación se denomina rol. Los roles pueden tener opcionalmente:

- Nombre.
- Expresión de multiplicidad.
- Navegabilidad.

La multiplicidad se presenta a continuación, y las otras dos características se discutirán más tarde.

La multiplicidad define cuántas instancias de una clase A pueden asociarse con una instancia de una clase B (ver Figura 6).

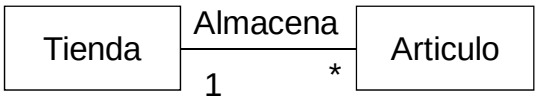

Figura 6. Multiplicidad de una asociación.

Por ejemplo, una instancia individual de una *Tienda* puede asociarse con "muchas" (cero o más. indicado por el "*"") instancias de *Articulo*.

En la Figura 7 se muestran algunos ejemplos de expresiones de multiplicidad.

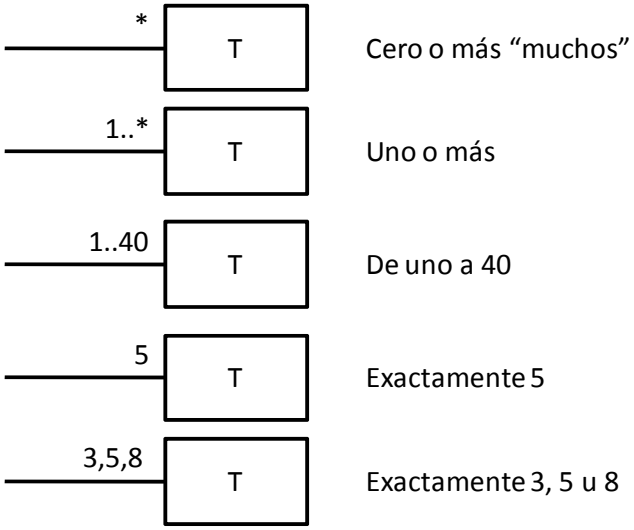

Figura 7. Multiplicidad de una asociación.

El valor de la multiplicidad indica cuántas instancias se puede asociar legalmente con otra, en un momento concreto, en lugar de a lo largo de un periodo de tiempo. Por ejemplo, es posible que un coche usado pudiera a lo largo del tiempo ser vendido repetidamente a comerciantes de coches usados. Pero en un momento concreto, el coche sólo es *Abastecido-por* **un** comerciante. El coche no es *Abastecido-por* **muchos** comerciantes en un momento concreto. Análogamente, en países con leyes monógamas, una persona puede estar *Casado-con* sólo **una** persona en un momento dado, aunque a lo largo del tiempo, puedan casarse con **muchas** personas.

El valor de la multiplicidad depende de nuestros intereses como modeladores y desarrolladores de software, porque pone de manifiesto una restricción de diseño que será (o podrá ser) reflejada en el software.

Rumbaugh proporciona otro ejemplo de *Persona* y *Compañia* en la asociación *Trabaja-para*. El que indiquemos si una instancia de Persona trabaja para una o varias instancias de Compañia, es dependiente del contexto del modelo; el departamento de impuestos está interesado en *muchas*; un sindicato probablemente sólo en *una*. Normalmente, la elección, prácticamente, depende de para quién estamos construyendo el software, y de esta manera, las multiplicidades válidas de una implementación

3.5. ¿Cómo de Detalladas Deben Ser las Asociaciones?

Las asociaciones son importantes, pero un error típico al crear los modelos del dominio, es dedicar demasiado tiempo durante el estudio intentando descubrirlas.

Es fundamental entender lo siguiente: Es más importante encontrar las clases conceptuales que las asociaciones. La mayoría del tiempo dedicado a la creación del modelo del dominio debería emplearse en la identificación de las clases conceptuales, no de las asociaciones.

3.6. Asignación de Nombres a las Asociaciones

Se recomienda nombrar una asociación en base al formato NombreTípo-FraseVerbal-NombreTípo donde la frase verbal crea una secuencia que es legible y tiene significado en el contexto del modelo.

Los nombres de las asociaciones deben comenzar con una letra mayúscula, puesto que una asociación representa un clasificador de enlace entre las instancias; en UML, los clasificadores deben comenzar con una letra mayúscula. Dos formatos típicos e igualmente válidos para un nombre de asociación compuesta son:

- Pagado-mediante.
- PagadoMediante.

La dirección por defecto para la lectura de los nombres de las asociaciones es de izquierda a derecha o de arriba abajo. No se corresponde con la dirección por defecto en UML, sino que se trata de una convención común.

3.7. Múltiples Asociaciones entre Dos Tipos

Dos tipos podrían tener múltiples asociaciones entre ellos; no es extraño. No existe ningún ejemplo destacado en nuestro caso de estudio del PDY, pero un ejemplo del dominio de la compañía aérea podría ser las relaciones entre un Vuelo (o quizás de manera más precisa, una EtapaVuelo) y un Aeropuerto (ver Figura 8); las asociaciones vuela-a y vuela-desde son relaciones claramente diferentes, que se deberían mostrar de manera separada.

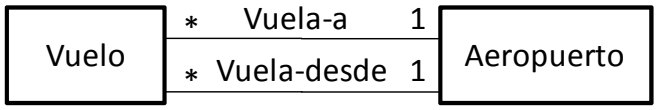

Figura 8. Múltiples asociaciones.

3.8. Asociaciones e Implementación

Durante el modelado del dominio, una asociación no es una declaración sobre el flujo de datos, variables de instancia o conexiones entre objetos en una solución software; es una manifestación de que una relación es significativa en un sentido puramente conceptual -en el mundo real-. Desde un punto de vista práctico, muchas de estas relaciones se implementarán generalmente en software como caminos de navegación y visibilidad (tanto en el Modelo de Diseño como en el Modelo de Datos), pero su presencia en una vista conceptual (o esencial) de un modelo de dominio no requiere su implementación.

Al crear un modelo de dominio, podríamos definir asociaciones que no son necesarias durante la implementación. A la inversa, podríamos descubrir asociaciones que necesitan implementarse pero que se obviaron durante el modelado del dominio. En estos casos, el modelo del dominio puede actualizarse para reflejar estos descubrimientos.

Posteriormente estudiaremos formas de implementar la asociaciones en un lenguaje de programación orientado a objetos (los más habitual es utilizar un atributo que referencia una instancia de la clase asociada), pero de momento, es conveniente pensar en ellas como

expresiones puramente conceptuales, no declaraciones sobre la solución de base de datos o software. Como siempre, al posponer las consideraciones de diseño nos libera de informaciones y decisiones extrañas mientras hacemos un estudio de "análisis" puro y maximiza nuestras opciones de diseño más adelante.

3.9. Asociaciones del Modelo del Dominio del PDV NuevaEra

Ahora podemos añadir las asociaciones a nuestro modelo del dominio del PDV Deberíamos añadir aquellas asociaciones que los requisitos (por ejemplo, los casos de uso) sugieren o implican una necesidad de recordar, o que, de otra manera, recomienda fuertemente nuestra percepción del dominio del problema. Cuando abordamos un nuevo problema, las categorías comunes de asociaciones, presentadas anteriormente, deberían revisarse y tenerse en cuenta, puesto que representan muchas de las asociaciones relevantes que normalmente necesitan recogerse.

La siguiente muestra de asociaciones es justificable en términos de necesito-conocer. Se basa en los casos de uso que se están considerando actualmente.

Registro Registra Venta	Para conocer la venta actual, generar el total, imprimir recibo
Venta Pagada-mediante Pago	Para conocer si se ha pagado la venta, relacionar la cantidad entregada con el total de la venta, e imprimir un recibo
CatalogoDeProductos Registra EspecificacionDelProducto	Para recuperar una *EspecificacionDelProducto* a partir de un articuloID

Recorreremos la lista de comprobación, basada en los tipos identificados previamente, considerando los requisitos del caso de uso actual.

Categoría	*Sistema*
A es una parte física de B	*Registro-Caja*
A es una parte lógica de B	*LineaDeVenta-Venta*
A está contenido físicamente en B	*Registo- Tienda* *Articulo- Tienda*
A está contenido iógicamente en B	*EspecificacionDeIProducto-Catalogo-DeProductos* *CatalogoDeProductos- Tienda*
A es una descnpción de 8	*EspecificacionDelProducto-Articulo*
A es una línea de una transacción o informe de B	*LineaDeVenta-Venta*
A se conoce/registra/recoge/informa/cap-tura en B	*(completa) Venta- Tienda* *(actual) Venta-Registro*
A es miembro de B	*Cajero-Tienda*
A es una subunidad organizativa de B	*no aplicable*
A utiliza o gestiona B	*Cajero-Registro*

	Encargado-Registro *Encargado-Cajero, aunque probablemente no es aplicable*
A se comunica con B	*Cliente-Cajero*
A está relacionado con una transacción B	*Cliente-Pago* *Cajero-Pago*
A es una transacción relacionada con otra transacción B	*Pago-Venta*
A está aliado de B	*LineaDeVenta-LineaDeVenta*
A es propiedad de B o informe de B	*Registro-Tienda*

3.10. ¿Conservamos Sólo las Asociaciones Necesito-Conocer?

El conjunto de asociaciones que aparecen anteriormente se derivaron de forma mecánica, la mayor parte, a partir de la lista de comprobación de asociaciones. Sin embargo, sería deseable que fuésemos más cuidadosos con las asociaciones que incluimos en nuestro modelo del dominio. Visto como una herramienta de comunicación, no es deseable sobrecargar el modelo del dominio con asociaciones que no se necesitan forzosamente y que no favorecen nuestra comprensión. Demasiadas asociaciones no imprescindibles oscurecen en lugar de aclarar.

Como se sugirió anteriormente, se recomienda centrarse en aquellas asociaciones para las que el conocimiento de la relación necesita conservarse durante algún tiempo (asociaciones "necesito-conocer"). Se recomienda evite mostrar asociaciones redundantes o derivadas.,

En base a este consejo, no todas las asociaciones que se acaban de mostrar son imprescindibles. Considérese lo siguiente:

Asociación	*Comentario*
Venta Insertada-por Cajero	Los requisitos no indican que necesito-conocer o registrar al cajero actual. Además, puede derivarse si está presente la asociación *Registro Usado-por Cajero.*
Registro Usado-por Cajero	Los requisitos no indican que necesito-conocer o registrar al -cajero actual.
Registro Iniciado-por Encargado	Los requisitos no indican que necesito-conocer o registrar al encargado que pone en marcha un *Registro.*
Venta Iniciada-por Cliente	Los requisitos no indican que necesito-conocer o registrar al cliente que inicia una venta.
Tienda Almacena Articulo	Los requisitos no indican que necesito-conocer o mantener información del inventario.
LineaDeVenta Registra-venta-de Articuio	Los requisitos no indican que necesito-conocer o mantener información del inventario.

Nótese que la capacidad de justificar una asociación en función de necesito-conocer depende de

los requisitos; obviamente, un cambio en éstos -como que se necesite mostrar en el recibo el ID del cajero- cambia la necesidad de recordar una relación.

Basado en el análisis anterior, podría justificarse la eliminación de las asociaciones en cuestión.

3.11.Asociaciones Necesito-Conocer vs. Comprensión

Un criterio necesito-conocer estricto para el mantenimiento de las asociaciones generará un "modelo de información" mínimo de lo que se necesita para modelar el dominio del problema -limitado por los requisitos actuales que se están considerando-. Sin embargo, este enfoque podría crear un modelo que no transmite (a nosotros o a los demás)1 una comprensión completa del dominio.

Además de ser un modelo necesito-conocer de información sobre las cosas, el modelo del dominio es una herramienta de comunicación con la que estamos intentando entender y comunicar a otros los conceptos importantes y sus relaciones. Desde este punto de vista, eliminando algunas asociaciones que no se exigen estrictamente en una base I necesito-conocer, puede crear un modelo sin interés -no comunica las ideas claves y relaciones-.

Por ejemplo, en la aplicación del PDV: aunque, tomando como base las relaciones necesito-conocer de manera estricta, podría no ser necesario registrar Venta Iniciada-por' Cliente, su ausencia deja fuera un aspecto importante para entender el dominio -que un cliente genera las ventas-.

En cuanto a las asociaciones, un buen modelo se sitúa en alguna parte entre un modelo necesito-conocer mínimo y uno que ilustra cada relación concebible. ¿El criterio básico para juzgar su valor? -¿Satisface todos los requisitos necesito-conocer y además comunica claramente un conocimiento esencial de los conceptos importantes en el dominio del problema?-.

Conviene centrarse en las asociaciones necesito-conocer, pero contemple las asociaciones de sólo-comprensión para enriquecer el conocimiento básico del dominio.

4. Atributos

Resulta útil identificar aquellos atributos de las clases conceptuales que se necesitan para satisfacer los requisitos de información de los actuales escenarios en estudio. Un **atributo** es un valor de datos lógico de un objeto.

Se deben incluir en un modelo del dominio aquellos atributos para los que los requisitos (o los casos de uso) sugieren o implican una necesidad de registrar la información.

Por ejemplo, un recibo (que recoge la información de una venta) normalmente incluye una fecha y una hora, y la dirección quiere conocer las fechas y horas de las ventas por múltiples motivos. En consecuencia, la clase conceptual *Venta* necesita los atributos *fecha* y *hora*.

En UML, los atributos se muestran en el segundo compartimento del rectángulo de la clase (Figura 9). Opcionalmente podrían mostrarse sus tipos.

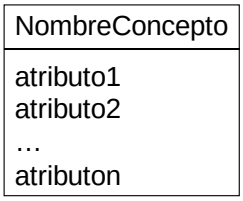

Figura 9. Clases y atributos.

4.1. Tipos de Atributos Válidos

Hay algunas cosas que no deberían representarse como atributos, sino como asociaciones. Esta sección presenta los tipos válidos.

Intuitivamente, la mayoría de los atributos simples son los que, a menudo, se conocen como tipos de datos primitivos, como los números. El tipo de un atributo, normalmente\ no debería ser un concepto de dominio complejo, como *Venta* o *Aeropuerto*.

Los atributos en un modelo del dominio deberían ser, preferiblemente, **atributos simples** o **tipos de datos**. Los tipos de datos de los atributos más comunes incluyen: Boolean, Fecha, Numero, String (Texto), Hora. Otros tipos comunes comprenden: Dirección, Color, Geométrico (Punto, Rectangulo), Numero de Teléfono, Numero de la Seguridad Social, Código de Producto Universal (del inglés UPC; Universal Product Code), códigos postales, tipos enumerados.

Un error típico es modelar un concepto del dominio complejo como un atributo. Para ilustrarlo, un aeropuerto de destino no es realmente una cadena de texto; se trata de una cosa compleja que ocupa muchos kilómetros cuadrados de espacio. Por tanto, *Vuelo* debería relacionarse con *Aeropuerto* mediante una asociación, no con un atributo.

Las clases conceptuales deben relacionarse mediante una asociación, no con un atributo.

La restricción de que el tipo de los atributos en el modelo del dominio sea sólo un tipo de datos

simple no implica que los atributos C+ + o lava (miembros de datos, campos de una instancia) sólo deban ser de tipos de datos primitivos, o simples. El modelo del dominio se centra en declaraciones conceptuales puras sobre un dominio del problema, no en componentes software. Posteriormente, durante el trabajo de diseño, se verá que las asociaciones entre objetos representadas en el modelo del dominio, a menudo, se implementarán como atributos que referencian a otros objetos software complejos. Sin embargo, ésta es sólo una de las posibles soluciones de diseño para implementar una asociación, y, de ahí que la decisión se deba posponer durante el modelado del dominio.

Los atributos deben ser, generalmente, tipos de datos. Esto es un término UML que implica un conjunto de valores para los cuales no es significativa una identidad única (en el contexto de nuestro modelo o sistema). Por ejemplo, no es (normalmente) significativo distinguir entre:

- Diferentes instancias del *Numero* 5.
- Diferentes instancias del *String* 'gato'.
- Diferentes instancias del *NumeroDeTelefono* que contiene el mismo número.
- Diferentes instancias de la *Direccion* que contiene la misma dirección.

Por el contrario, es significativo distinguir (por identidad) entre dos instancias de *Persona* cuyos nombres son, en los dos casos, "Luis García", puesto que las dos instancias pueden representar individuos diferentes con el mismo nombre.

Por tanto, todos los tipos primitivos (número, string) son tipos de datos UML, pero no todos los tipos de datos son primitivos. Por ejemplo, *NumeroDeTelefono* es un tipo de datos no primitivo.

La noción de tipos de datos puede ser sutil. Como regla empírica, se recomienda ser fieles a la prueba básica de tipos de atributos "simples": se pondrá como un atributo si se considera de manera natural como un número, string, booleano, fecha u hora (etcétera); en otro caso, se representará como una clase conceptual aparte. En caso de duda, definir algo como una clase conceptual aparte en lugar de como un atributo.

4.2. Clases de Tipos de Datos no Primitivos

El tipo de un atributo podría representarse como una clase no primitiva por derecho propio en un modelo del dominio. Por ejemplo, en el sistema de PDV hay un identificador de artículo. Generalmente se ve simplemente como un número. Así que, ¿se debería representar como una clase no primitiva? Se recomienda aplicar esta guía. Representar lo que podría considerarse, inicialmente, como un tipo de dato primitivo (como un número o string) como una clase no primitiva si:

- Está compuesto de secciones separadas. Por ejemplo, un número de teléfono y el nombre de una persona.
- Habitualmente, hay operaciones asociadas con él, como análisis sintáctico o validación. Por ejemplo, el DNI.
- Tiene otros atributos. Por ejemplo, el precio de una promoción podría tener una fecha

(efectiva) de comienzo y fin.

- Es una cantidad con una unidad. Por ejemplo, la cantidad del pago tiene una unidad monetaria.

- Es una abstracción de uno o más tipos con alguna de estas cualidades. Por ejemplo, el identificador del artículo en el dominio de ventas es una generalización de tipos como el Código de Producto Universal (UPC) o el Número de Artículo Europeo (EAN)

Aplicando estas guías a los atributos del modelo del dominio del PDV se llega al siguiente análisis:

- El identificador del artículo es una abstracción de varios esquemas de codificación comunes, incluyendo UPC-A, UPC-E y la familia de esquemas EAN. Estos esquemas de codificación numéricos tienen subpartes que identifican al fabricante, producto, país (para EAN), y un dígito de control para validarlos. Por tanto, debería existir una clase no primitiva *ArticuloID*, puesto que satisface muchas de las guías anteriores.

- Los atributos del *precio* y *cantidad* deberían ser clases no primitivas *Cantidad* o *Moneda* porque se trata de cantidades en una unidad monetaria.

- El atribulo de la dirección debe ser una clase no primitiva *Dirección* porque tiene secciones diferentes.

Las clases *ArticuloID*, *Direccion* y *Cantidad* son tipos de datos (no es significativa la identidad única de las instancias) pero merece la pena considerarlas como clases independientes debido a sus cualidades.

¿Debería mostrarse la clase *ArticuloID* como una clase conceptual independiente en el modelo del dominio? Depende de lo que quiera resaltar en el diagrama. Puesto que *ArticuloID* es un *tipo de datos* (no es importante la identidad única de las instancias), se podría mostrar en el compartimento de los atributos del rectángulo de la clase. Pero, puesto que es una clase no primitiva, con sus propios atributos y asociaciones, podría ser interesante mostrarla como una clase conceptual en su propio rectángulo. No existe una respuesta correcta; depende de cómo se esté utilizando el modelo del dominio como herramienta de comunicación, y la importancia de los conceptos en el dominio. Un modelo del dominio es una herramienta de comunicación; las elecciones sobre lo quel se muestra deben hacerse con esa consideración en mente.

4.3. Deslizarse al Diseño: Ningún Atributo Como Clave Ajena

No se deberían utilizar los atributos para relacionar las clases conceptuales en el modelo del dominio. La violación más típica de este principio es añadir un tipo de atributo de clave ajena, como se hace normalmente en el diseño de bases de datos relacionales, para asociar dos tipos. Por ejemplo, en la Figura 10, no es deseable el atributo *numeroRegistroActual* en la clases *Cajero* porque el propósito es relacionar el *Cajero* con un objeto *Registro*. La mejor manera de expresar que un *Cajero* utiliza un *Registro* es con una asociación, no con un atributo de clave ajena.

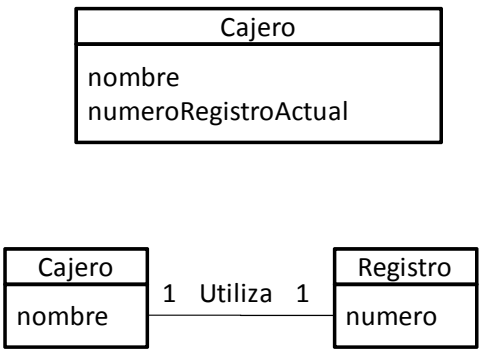

Figura 10. No utilizar atributos como claves ajenas.

Hay muchas formas de relacionar objetos -siendo las claves ajenas una de ella y pospondremos el modo de implementar la relación hasta el diseño, para evitar un deslizamiento al diseño.

4.4. Modelado de Cantidades y Unidades de los Atributos

La mayoría de las cantidades numéricas no deberían representarse simplemente como números. Consideremos el precio o la velocidad. Son cantidades con unidades asociadas, y, es habitual que se necesite conocer la unidad y dar soporte a las conversiones. El software del PDV NuevaEra es para un mercado internacional y necesita soportar los precios en diferentes monedas. En el caso general, la solución consiste en representar la *Cantidad* como una clase conceptual aparte, con una *Unidad* asociada. Puesto que las cantidades se consideran tipos de datos (no es importante la identidad única de las instancias), es aceptable recoger su representación en la sección de atributos del rectángulo de clase (ver Figura 11). También es común mostrar especializaciones de *Cantidad*. El *Dinero* es una clase de *Cantidad* cuyas unidades son monedas. El *Peso* una cantidad cuyas unidades son kilogramos o libras.

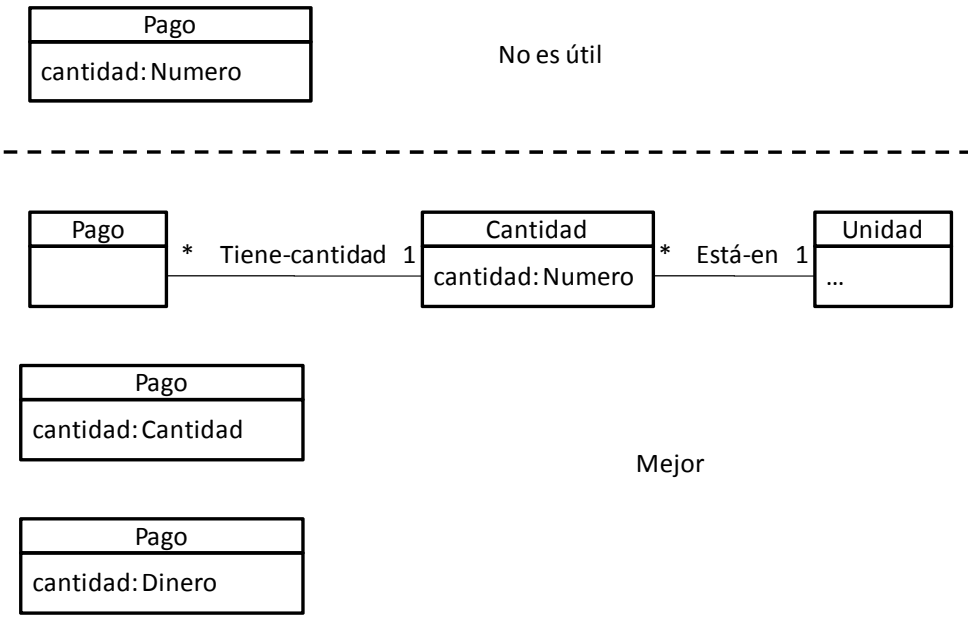

Figura 11. Modelado de cantidades.

4.5. Atributos en el Modelo del Dominio de NuevaEra

Los atributos elegidos reflejan los requisitos de esta iteración -los escenarios de Procesar Venta de esta iteración-.

CONCEPTO	*ATRIBUTO*
Pago	*cantidad*: Se debe capturar una cantidad (también conocida como "cantidad entregada") para determinar si se proporciona el pago suficiente y calcular el cambio.
Especificacion Del Producto	*descripcion*: Para mostrar la descripción en una pantalla o recibo.
	id: Para buscar una *EspecificacionDelProducto,* dado un articuloID introducido, es necesario relacionadas con un *id*.
	precio: Para calcular el total de la venta y mostrar el precio de la línea de venta.
Venta	*fecha, hora*: Un recibo es un informe en papel de una venta. Normalmente muestra la fecha y la hora de la venta.
LineaDeVenta:	*cantidad*: Para registrar la cantidad introducida, cuando hay más de un artículo en una línea de venta (por ejemplo, cinco paquetes de tofu).
Tienda tienda.	*direccion. nombre*: El recibo requiere el nombre y la dirección de la

4.6. Atributos Derivados

Es posible que el cajero reciba un grupo de artículos iguales (por ejemplo, seis paquetes de tofu), introduzca el *articuloID* una vez, y después introduzca una cantidad (por ejemplo, seis). En consecuencia, una *LineaDeVenta* individual se puede asociar con mál de una instancia de un artículo.

La cantidad introducida por el cajero podría registrarse como un atributo de la *LineaDeVenta*. Sin embargo, la cantidad se puede calcular a partir del valor de la multiplicidad actual de la relación, de ahí que pudiera caracterizarse como un atributo derivado -uno que puede derivarse a partir de otra información-. En UML, un atributo derivado se indica con el símbolo "/".

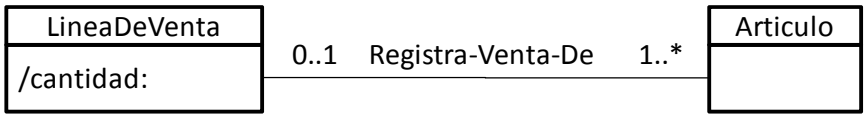

Figura 12. Atributos derivados.

5. Conclusión del Modelo del Dominio

La combinación de las clases conceptuales, asociaciones y atributos, descubiertos en el estudio anterior, da lugar al modelo que se presenta en la Figura 13.

Se ha creado un modelo del dominio, relativamente útil, para el dominio de una aplicación de PDV. No existe un único modelo correcto. Todos los modelos son aproximaciones del dominio que estamos intentando entender. Un buen modelo del dominio captura las abstracciones y la información esenciales necesarias para entender el dominio en el contexto de los requisitos actuales, y ayuda a la gente a entender el dominio -sus conceptos, terminología y relaciones-.

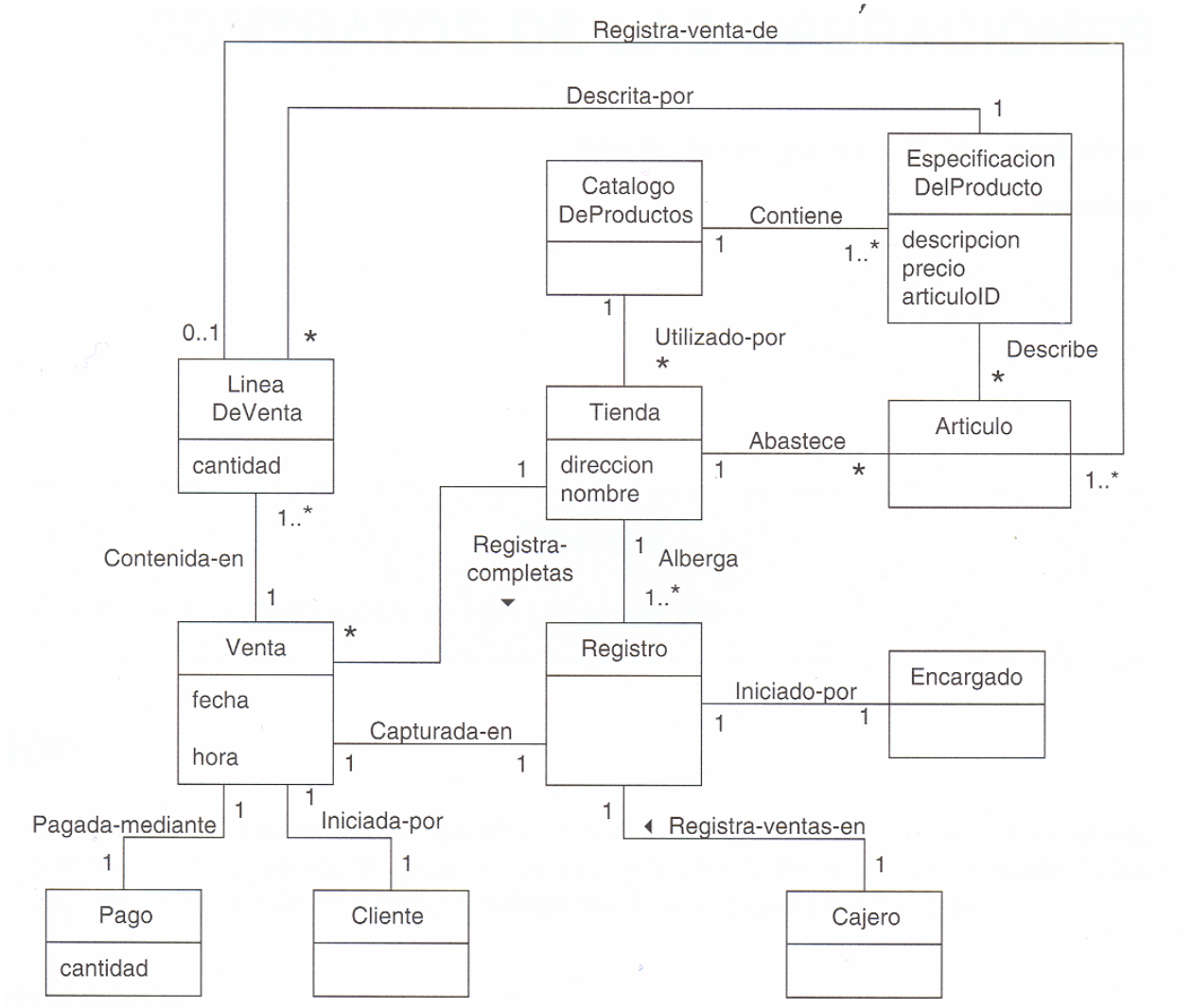

Figura 13. Un modelo del dominio parcial.