9

CONSTRUCCIÓN DE UN MODELO CONCEPTUAL

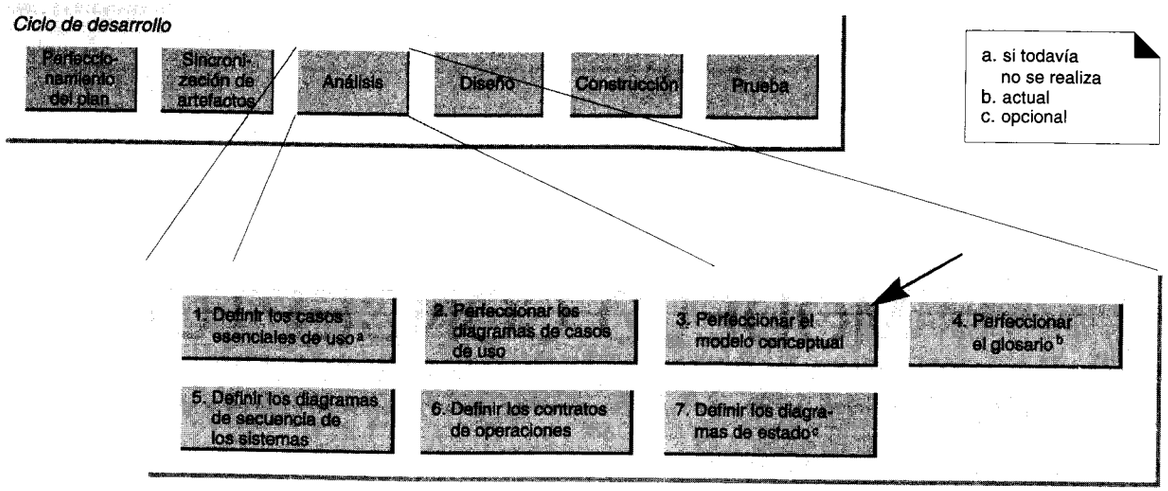
Objetivos

- Identificar los conceptos relacionados con los requerimientos del ciclo actual de desarrollo.
- Crear un modelo conceptual preliminar.
- Distinguir entre los atributos correctos y los incorrectos.
- Incorporar conceptos de *especificación* cuando convenga.
- Comparar los términos: concepto, tipo, interfaz y clase.

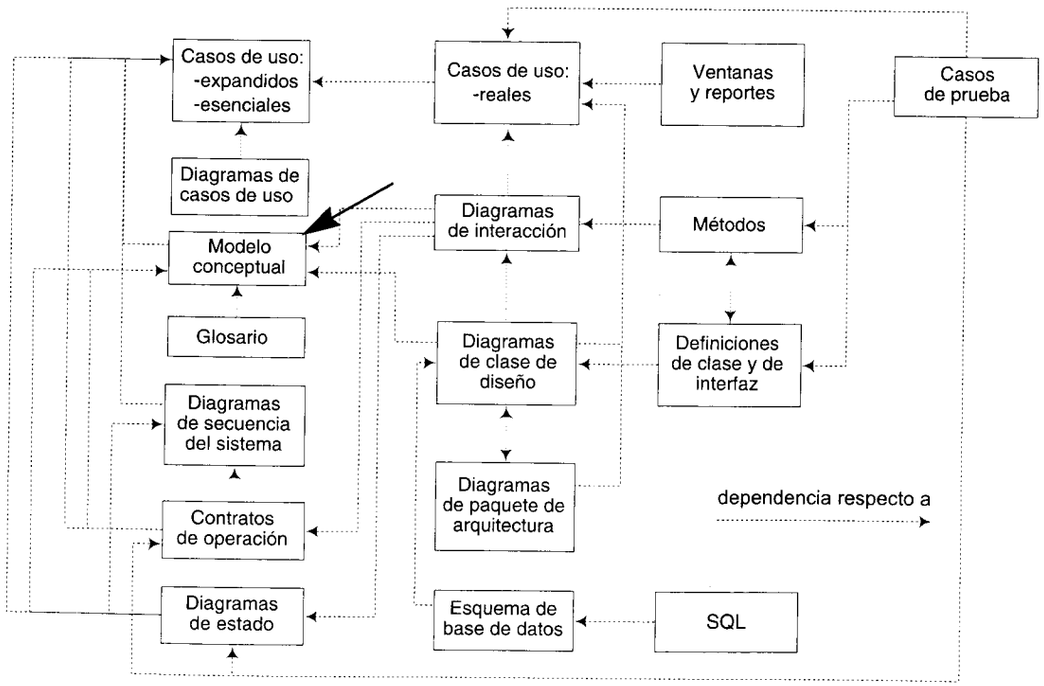
9.1 Introducción

Un modelo conceptual explica (a sus creadores) los conceptos significativos en un dominio del problema; es el artefacto más importante a crear durante el análisis orientado a objetos.[1] En este capítulo estudiaremos los conocimientos preliminares en la creación de modelos conceptuales. En los dos siguientes examinaremos más detenidamente las habilidades relacionadas con la construcción de modelos conceptuales: observar atentamente los atributos y las asociaciones.

[1] Los casos de uso son un importante artefacto del análisis de requerimientos, pero realmente no están orientados a *objetos*. Ponen de relieve la vista del dominio a partir de un proceso.



Actividades de la fase de análisis dentro de un ciclo de desarrollo.



Dependencias de los artefactos durante la fase de construcción.

Identificar muchos objetos o conceptos constituye la esencia del análisis orientado a objetos, y el esfuerzo se compensa con los resultados conseguidos durante la fase de diseño e implementación.

La identificación de conceptos forma parte de una investigación del dominio del problema. El lenguaje UML contiene la notación en diagramas de estructura estática que explican gráficamente los modelos conceptuales.

> Una cualidad esencial que debe ofrecer un modelo conceptual es que representa cosas del mundo real, no componentes del software.

9.2 Actividades y dependencias

Una de las primeras actividades centrales de un ciclo de desarrollo consiste en crear un modelo conceptual para los casos de uso del ciclo actual. Esto no puede hacerse si no se cuentan con los casos y con otros documentos que permitan identificar los conceptos (objetos). La creación no siempre es lineal; por ejemplo, el modelo conceptual puede formularse en paralelo con el desarrollo de los casos.

9.3 Modelos conceptuales

El paso esencial de un análisis o investigación orientados a *objetos* es descomponer el problema en conceptos u objetos individuales: las cosas que sabemos. Un **modelo conceptual** es una representación de conceptos en un dominio del problema [MO95, Fowler96]. En el UML, lo ilustramos con un grupo de **diagramas de estructura estática** donde no se define ninguna operación. La designación de *modelo conceptual* ofrece la ventaja de subrayar fuertemente una concentración en los conceptos del dominio, no en las entidades del software.

Puede mostrarnos:

- conceptos
- asociaciones entre conceptos
- atributos de conceptos

Por ejemplo, en la figura 9.1 vemos un modelo conceptual parcial del dominio de la tienda y las ventas. Explica gráficamente que el concepto de *Pago* y *Venta* son importantes en este dominio del problema, que *Pago* se relaciona con *Venta* en una forma que conviene señalar y que *Venta* tiene fecha y hora. Por ahora no son importantes para nosotros los detalles de la notación.

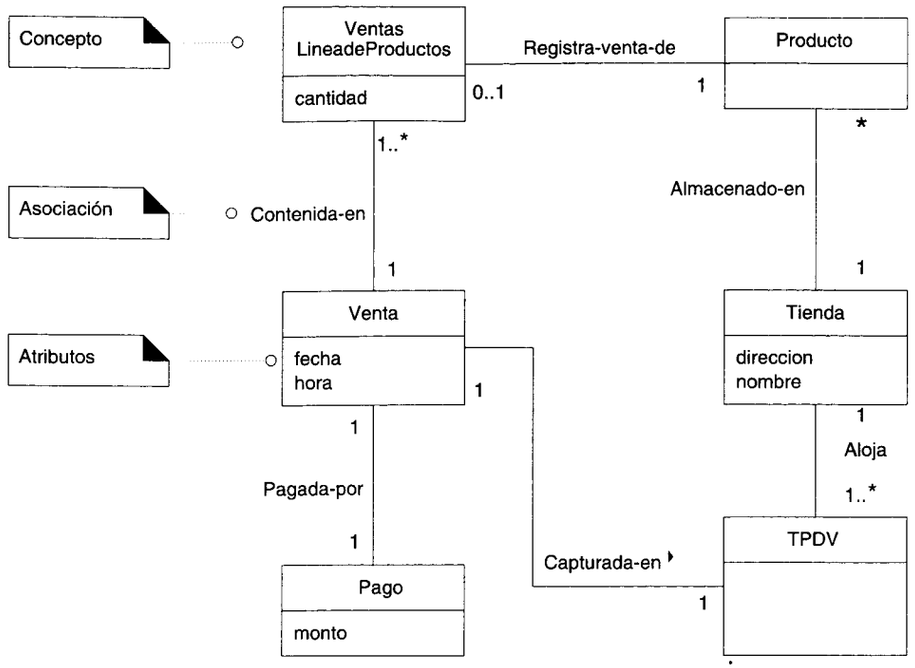


Figura 9.1 Modelo conceptual parcial. Los números en los extremos de la línea indica multiplicidad, la cual se describe en un capítulo subsecuente.

9.3.1 Conocimiento de la nomenclatura del dominio

Además de descomponer el espacio del problema en unidades comprensibles (conceptos), la creación de un modelo conceptual contribuye a esclarecer la terminología o nomenclatura del dominio. Podemos verlo como un modelo que comunica (a los interesados como pueden serlo los desarrolladores) cuáles son los términos importantes y cómo se relacionan entre sí.

9.3.2 Los modelos conceptuales no son modelos de diseño de software

Un modelo conceptual, como se advierte en la figura 9.2, es una descripción del dominio de un problema real, *no* es una descripción del diseño del software, como una clase de Java o de C++ (figura 9.3). Por ello los siguientes elementos no son adecuados en él:

- Los artefactos de software, como una ventana o una base de datos, salvo que el dominio a modelar se refiera a conceptos de software; por ejemplo, un modelo de interfaces gráficas para el usuario.

- Las responsabilidades o métodos.[1]

Figura 9.2 Un modelo conceptual muestra conceptos del mundo real.

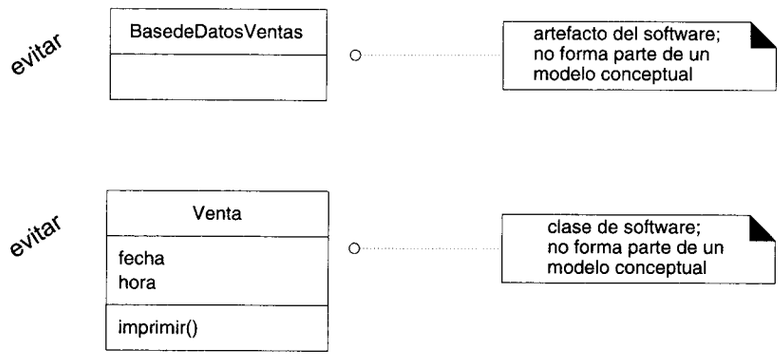


Figura 9.3 Un modelo conceptual no muestra los artefactos o clases del software.

9.3.3 Conceptos

En términos informales el concepto es una idea, cosa u objeto. En un lenguaje más formal, podemos considerarlo a partir de su símbolo, intensión[2] y extensión [MO95] (figura 9.4).

- **Símbolo**: palabras o imágenes que representan un concepto.
- **Intensión**: la definición del concepto.
- **Extensión**: el conjunto de ejemplos a que se aplica el concepto.

Consideremos, por ejemplo, el concepto del evento de una transacción de compra. Podemos optar por designarlo con el símbolo *Venta*. La intensión de una *Venta* puede afirmar que "representa el evento de una transacción de compra y tiene fecha y hora". La extensión de *Venta* son todos los ejemplos de venta; en otras palabras, el conjunto de todas las ventas.

---

[1] Las responsabilidades normalmente se relacionan con entidades del software y los métodos siempre lo hacen; pero el modelo conceptual describe conceptos reales, no entidades del software. Durante la fase de *diseño* es muy importante tener en cuenta las responsabilidades; ya que no sólo forman parte de este modelo.

[2] Intensión: en oposición a extensión, designa el grado de una cualidad.

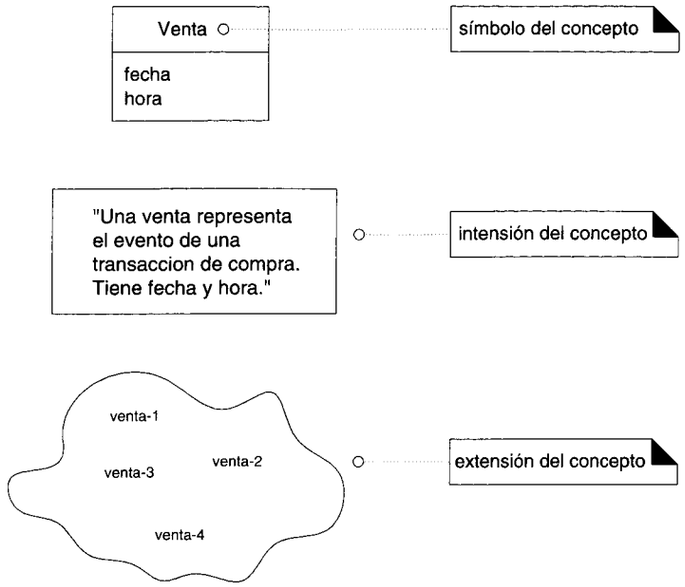


Figura 9.4 El concepto tiene un símbolo, intensión y extensión.

Cuando se crea un modelo conceptual, por lo regular la vista del símbolo y de la intensión de un concepto es el aspecto de mayor interés práctico.

9.3.4 Los modelos conceptuales y la descomposición

Los problemas de software a veces son complejos; la descomposición —divide y vencerás— es una estrategia que suele utilizarse para resolver la complejidad dividiendo el espacio del problema en unidades comprensibles. En el **análisis estructurado** la dimensión de la descomposición se realiza mediante procesos o *funciones*. En cambio, en el análisis orientado a objetos, se lleva a cabo fundamentalmente con conceptos.

> Una distinción fundamental entre el análisis orientado a objetos y el análisis estructurado: división por conceptos (objetos) y no por funciones.

Por tanto, una tarea primordial de la fase de análisis consiste en identificar varios conceptos en el dominio del problema y documentar los resultados en un modelo conceptual.

9.3.5 Conceptos en el dominio del punto de venta

Por ejemplo, en el dominio del problema real de comprar productos en una tienda en una terminal de punto de venta (TPDV) intervienen los conceptos de *Tienda, TPDV* y una *Venta*. Por tanto, nuestro modelo conceptual (figura 9.5) puede incluir una *Tienda, TPDV* y una *Venta*.

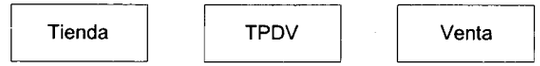


Figura 9.5 Modelo conceptual parcial en el dominio de la tienda.

9.4 Estrategias para identificar los conceptos

Nuestra meta es crear un modelo conceptual de conceptos interesantes o significativos del dominio en cuestión. En este caso, ello significa conceptos relacionados con el caso de uso *Comprar productos, versión 1.* La tarea fundamental será, pues, identificar los conceptos; se proponen dos estrategias.

La siguiente es una directriz de gran utilidad en la identificación de conceptos:

> Es mejor exagerar y especificar un modelo conceptual con muchos conceptos refinados que no especificarlo cabalmente.

No piense que un modelo conceptual es más adecuado si tiene menos conceptos; generalmente suele suceder lo contrario.

Es frecuente omitir conceptos durante la fase inicial de identificación y descubrirlos más tarde cuando se examinen los atributos o asociaciones o durante la fase de diseño. Cuando se detecten, habrá que incorporarlos al modelo conceptual.

No se excluya un concepto simplemente porque los requerimientos no indiquen una necesidad evidente que permita recordar la información acerca de ella (criterio común de la construcción de modelos de datos para diseñar una base de datos relacional, pero no pertinente a la creación de modelos conceptuales) o porque el concepto carezca de atributos. Es perfectamente válido tener conceptos sin atributos o conceptos con un papel puramente de comportamientos en el dominio en vez de un papel informacional.

9.4.1 Obtención de conceptos a partir de una lista de categorías de conceptos

La creación de un modelo conceptual se comienza preparando una lista de conceptos idóneos a partir de la siguiente lista. Contiene muchas categorías comunes que vale la pena tener en cuenta, sin que importe el orden de importancia. Los ejemplos se tomaron de los dominios de la tienda y de las reservaciones de líneas aéreas.

Categoría del concepto	Ejemplos
objetos físicos o tangibles	*TPDV* *Avion*
especificaciones, diseño o descripciones de cosas	*EspecificaciondeProducto* *DescripciondeVuelo*
lugares	*Tienda* *Aeropuerto*
transacciones	*Venta, Pago* *Reservacion*
línea o renglón de elemento de transacciones	*VentasLineadeProducto*
papel de las personas	*Cajero* *Piloto*
contenedores de otras cosas	*Tienda, Cesto* *Avion*
cosas dentro de un contenedor	*Producto* *Pasajero*
otros sistemas de cómputo o electromecánicos externos al sistema	*SistemadeAutorizaciondeTarjetadeCredito* *ControldeTraficoAereo*
conceptos de nombres abstractos	*Hambre* *Acrofobia*
organizaciones	*DepartamentodeVentas* *ObjetoLineaAerea*
eventos	*Venta, Robo, Junta* *Vuelo, Accidente, Aterrizaje*

Categoría del concepto	Ejemplos
procesos (a menudo *no* están representados como conceptos, pero pueden estarlo)	*VentaUnProducto* *ReservacionAsiento*
reglas y políticas	*PoliticadeReembolso* *PoliticadeCancelaciones*
catálogos	*CatalogodeProducto* *CatalogodePartes*
registros de finanzas, de trabajo, de contratos de asuntos legales	*Recibo, Mayor, ContratodeEmpleo* *BitacoradeMantenimiento*
instrumentos y servicios financieros	*LineadeCredito* *Existencia*
manuales, libros	*ManualdePersonal* *ManualdeReparaciones*

9.4.2 *Obtención de conceptos a partir de la identificación de frases nominales*

Otra técnica muy útil (por su simplicidad) propuesta en [Abbot83] consiste en identificar las frases nominales en las descripciones textuales del dominio de un problema y considerarlas conceptos o atributos idóneos.

> Este método hay que utilizarlo con mucha prudencia; no es posible encontrar mecánicamente correspondencias entre sustantivo y concepto, y además las palabras del lenguaje natural son ambiguas.

Pese a ello, esta técnica es fuente de inspiración. Los casos expandidos de uso son una excelente descripción que puede conseguirse con este análisis. Por ejemplo, puede usarse el caso de uso *Comprar productos, versión 1.*

Acción de los actores	Respuesta del sistema
1. Este caso de uso comienza cuando un **Cliente** llega a una **caja de TPDV** con **productos** que desea comprar.	
2. El **Cajero** registra el **código universal de productos** (CUP) en cada **producto**. Si hay más de un **producto**, el **Cajero** puede introducir también la **cantidad**.	3. Determina el **precio del producto** y a la **transacción de ventas** le agrega la información sobre el producto. Se muestran la **descripción** y el **precio** del **producto** actual.

Algunas de las frases nominales anteriores son conceptos idóneos; algunas pueden ser atributos de conceptos. Por favor, consulte el lector la sección siguiente y el capítulo dedicado a los atributos: en ellos encontrará sugerencias para distinguirlos.

Una debilidad de este enfoque es la imprecisión del lenguaje natural; varias frases nominales pueden designar el mismo concepto o atributo, entre otras ambigüedades que pueden presentarse. Pese a ello, no dudamos en recomendar usarlo en combinación con el método de *Lista de categoría de conceptos.*

9.5 Conceptos idóneos para el dominio del punto de venta

A partir de la *Lista de categoría de conceptos* y del análisis de frases nominales generamos una lista de conceptos adecuados para incluirlos en la aplicación del punto de venta. La lista está sujeta a la restricción de los requerimientos y simplificaciones que se consideren en el momento: los casos simplificados de uso de *Comprar productos, versión 1.*

TPDV
Producto
Tienda
Venta
Pago
CatálogodeProductos
EspecificaciondeProducto
VentasLineadeProductos
Cajero
Cliente
Gerente

9.5.1 Objetos del informe: ¿se incluye el recibo en el modelo?

El recibo es un registro de una venta y de un pago, así como un concepto relativamente prominente en el dominio de ventas; ¿debe, pues, mostrarse en el modelo? En seguida se mencionan algunos factores que han de tenerse presentes:

- El recibo es un informe de una venta. En general, no conviene incluirlo en un modelo conceptual, ya que toda su información proviene de otras fuentes. Éste es un buen motivo para excluirlo.
- El recibo cumple un papel especial respecto a las reglas de la empresa: al portador le confiere el derecho de devolver los productos adquiridos. Esta es una razón para incorporarlo al modelo.

El recibo se excluirá, porque las devoluciones de productos no se incluyen en este ciclo de desarrollo. Se justificará su inclusión durante el ciclo que aborde el caso *Devolver productos.*

9.5.2 El modelo conceptual del punto de venta (sólo conceptos)

La lista anterior de los nombres de conceptos puede representarse gráficamente (figura 9.6) en la notación del diagrama de estructura estática de UML, a fin de mostrar la génesis del modelo conceptual.

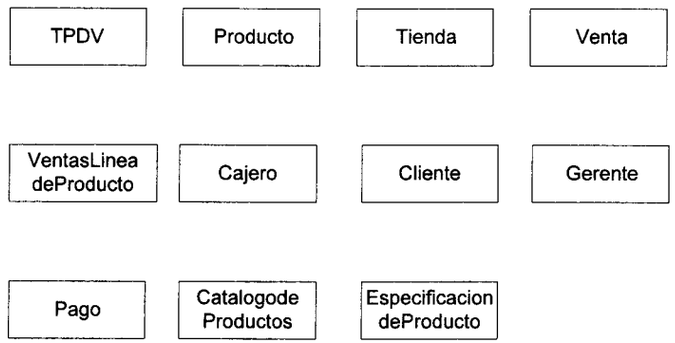


Figura 9.6 Modelo conceptual inicial del dominio del punto de venta.

En capítulos posteriores trataremos de los atributos y asociaciones del modelo conceptual.

9.6 Directrices para construir modelos conceptuales

9.6.1 *Cómo construir un modelo conceptual*

Aplique los siguientes pasos para crear un modelo conceptual:

> Para construir un modelo conceptual:
>
> 1. Liste los conceptos idóneos usando la *Lista de categorías de conceptos* y la identificación de la frase nominal relacionadas con los requerimientos en cuestión.
> 2. Dibújelos en un modelo conceptual.
> 3. Incorpore las asociaciones necesarias para registrar las relaciones para las cuales debe reservar un espacio en la memoria (tema que se expondrá en un capítulo posterior).
> 4. Agregue los atributos necesarios para cumplir con las necesidades de información (tema que se tratará en un capítulo posterior).

9.6.2 *La asignación de nombres y el modelado de las cosas: el cartógrafo*

La estrategia del cartógrafo se aplica a los mapas y a los modelos conceptuales.

> Prepare un modelo conceptual inspirándose en la metodología del cartógrafo:
>
> - Utilice los nombres existentes en el territorio.
> - Excluya las características irrelevantes.
> - No agregue cosas que no existan.

El modelo conceptual es una especie de mapa de conceptos o cosas de un dominio. Este enfoque pone de relieve el papel analítico de un modelo conceptual y sugiere lo siguiente:

- Los cartógrafos se sirven de nombres del territorio —no cambian los nombres de ciudades en sus mapas. En el caso de un modelo conceptual, ello significa *utilizar el vocabulario del dominio cuando se asignan nombres a los conceptos y a los atributos.* Por ejemplo, al desarrollar el modelo de una biblioteca, al cliente se le designará con los nombres que utilice el personal: *Visitante, Lector* u otro semejante.

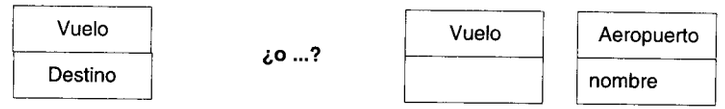
- Un cartógrafo elimina cosas en el mapa en caso de que no las juzgue pertinentes para el propósito que persigue; así, no es necesario que muestre la topografía ni las poblaciones. De modo análogo, un modelo conceptual puede excluir en el dominio del problema los conceptos que no se relacionen con los requerimientos. Por ejemplo, podemos omitir *Pluma* y *BolsadePapel* en nuestro modelo conceptual (con el conjunto actual de requerimientos), por no tener una función importante que sea obvia.
- Un cartógrafo no muestra cosas que no existan; por ejemplo, una montaña inexistente. En forma parecida, el modelo conceptual ha de excluir las cosas que *no* se encuentren en el dominio del problema en cuestión.

9.6.3 *Un error que se comete frecuentemente al identificar los conceptos*

Tal vez el error más frecuente cuando se crea un modelo conceptual es el de representar algo como atributo, cuando debió haber sido un concepto. Una regla práctica para no caer en él es:

> Si en el mundo real no consideramos algún concepto X como número o texto, probablemente X sea un concepto y no un atributo.

Por ejemplo, pongamos el caso del dominio de las reservaciones en líneas aéreas. ¿Debería *Destino* ser un atributo de *Vuelo* o un concepto aparte *Aeropuerto*?



En el mundo real, un aeropuerto de destino no se considera número ni texto: es una cosa masiva que ocupa espacio. Por tanto, *Aeropuerto* debería ser un concepto.

> En caso de duda, convierta el atributo en un concepto independiente.

9.7 Solución de los conceptos similares: comparación entre TPDV y Registro

Antaño, mucho antes del advenimiento de las terminales instaladas en el punto de venta, una tienda llevaba un *registro*: un libro donde asentaba las ventas y los pagos. Con el tiempo fue automatizado en un registro mecánico de efectivo. Hoy esa terminal desempeña el papel del registro (figura 9.7).

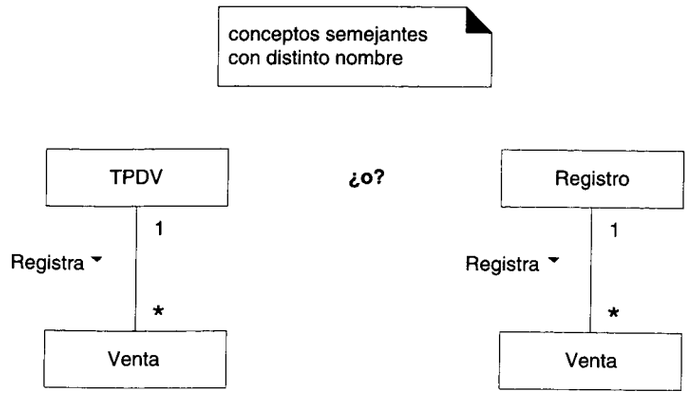


Figura 9.7 TPDV y registro son conceptos similares.

Un registro es una cosa que asienta las ventas y los pagos, pero también lo es la terminal instalada en el punto de ventas. No obstante, el término *registro* parece tener un significado más abstracto y denota una implementación menos orientada que *TPDV.* Pues bien, ¿en el modelo conceptual deberíamos utilizar el símbolo *Registro* en vez de *TPDV*?

> Primero, una regla práctica consiste en que un modelo conceptual no es absolutamente correcto ni erróneo, sino de mayor o menor utilidad; es una herramienta de la comunicación.

Conforme al principio del cartógrafo, *TPDV* es un término usual en el territorio, por lo cual es un símbolo útil desde el punto de vista del conocimiento y la comunicación. *Registro* es atractivo y útil, según la meta de crear modelos que representen abstracciones y que no dependan de la implementación.[1]

Ambas opciones tienen sus bondades. En este caso de estudio hemos escogido *TPDV* de modo un tanto arbitrario; también pudimos haber escogido *Registro.*

9.8 Construcción de un modelo del mundo *irreal*

Algunos sistemas de software están destinados a dominios que presentan muy poca semejanza con los dominios naturales o con los de las empresas; un ejemplo lo constituye el software de las telecomunicaciones. Todavía es posible crear un modelo conceptual en esos dominios, pero se requiere un alto grado de abstracción y abandonar los diseños comunes.

[1] Nótese que antaño un *registro* era simplemente una implementación posible de la manera de registrar las ventas. Con el tiempo, su significado ha ido generalizándose.

Enumeramos algunos conceptos adecuados que se relacionan con un intercambio de telecomunicaciones: *Mensaje, Conexión, Diálogo, Ruta, Protocolo.*

9.9 Especificación o descripción de conceptos

Suponga lo siguiente:

- La instancia de *Elemento* representa una entidad física de una tienda; puede ser incluso un número serial.
- Un *Elemento* tiene una descripción, precio y código universal de producto, los cuales no están registrados en ninguna otra parte.
- Todos los que trabajan en la tienda sufren amnesia.
- Cada vez que se vende un elemento físico, en "terreno del software" se elimina una instancia del software correspondiente a *Elemento.*

Con estas suposiciones, preguntamos: ¿qué sucede en el siguiente escenario?

Existe gran demanda de una nueva hamburguesa: ObjetoHamburguesa. La tienda vende todas sus existencias, lo cual significa que todas las instancias de *Elemento* de ObjetoHamburguesa se cancelan en la memoria de la computadora.

Ahora bien, ésta es la esencia del problema: si alguien pregunta: "¿Cuánto cuesta el ObjetoHamburguesa?", nadie podrá contestarle porque la memoria de su precio se anexó a las instancias inventariadas, que fueron eliminándose conforme se vendían.

Nótese asimismo que el modelo actual, si se implementa en el software tal como se describe, posee datos duplicados y maneja ineficientemente el espacio, porque la descripción, el precio y el código universal de producto se duplica en cada instancia de *Elemento* del mismo producto.

9.9.1 *Necesidad de las especificaciones*

El problema anterior demuestra la necesidad de un concepto de objetos que son especificaciones o descripciones de otras cosas. Para resolver el problema del *Elemento* lo que se necesita es una *EspecificaciondeProducto* (o *EspecificaciondeElemento, DescripciondeProducto, ...*) concepto que registra la información sobre los elementos. Una *EspecificaciondeProducto* no representa un *Elemento,* sino una descripción *acerca de* ellos. Nótese que, aunque todos los elementos inventariados se vendan y se eliminen sus instancias correspondiente de software, se conserva la *EspecificaciondeProducto.*

La descripción o especificación de objetos se relacionan bastante con aquella que describen. En un modelo conceptual, se acostumbra estipular que una *EspecificacionX describe una X.*

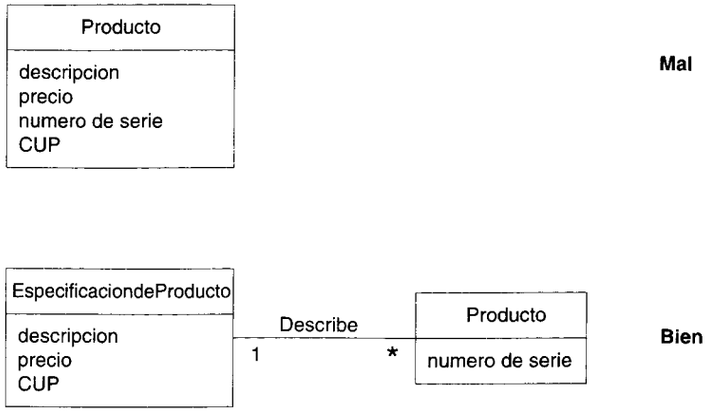


Figura 9.8 Especificaciones de otras cosas. El signo "*" significa una multiplicidad de "muchos". Indica que una *EspecificacióndeProducto* puede describir muchos (*) *Productos.*

La necesidad de especificar los conceptos es frecuente en los dominios de ventas y productos. También lo es en la manufactura, donde se requiere una *descripción* de lo manufacturado que se distingue de la cosa manufacturada. El tiempo y el espacio han sido incluidos al explicar la causa de la especificación de conceptos, por ser muy comunes; no se trata de un concepto poco usual de la construcción de modelos.

9.9.2 ¿Cuándo se requiere especificar los conceptos?

Las siguientes directrices indican cuándo emplear las especificaciones.

> Incorpore una especificación o descripción de conceptos (por ejemplo, *EspecificacióndeProducto*) cuando:
>
> - La eliminación de las instancias de las cosas que describen *Elemento*, por ejemplo, da por resultado una pérdida de la información que ha de conservarse, debido a la asociación incorrecta de la información con lo eliminado.
> - Reduce información redundante o duplicada.

9.9.3 Otro ejemplo de especificación

He aquí un último ejemplo: imagine que una compañía aérea sufre el accidente fatal de uno de sus aviones. Suponga que todos los vuelos quedan cancelados por seis meses, mientras se lleva a cabo la investigación. Suponga además que, cuando se cancelan los vuelos, sus correspondientes objetos de software *Vuelo* se eliminan en la memoria de la computadora. Así, todos los objetos *Vuelo* quedan eliminados tras el accidente.

Si el único registro del aeropuerto al cual se dirige un vuelo está en las instancias *Vuelo*, que representan vuelos específicos en determinado día y hora, ya no habrá un registro de qué rutas tiene la línea aérea.

Para resolver este problema, se requiere una *DescripciondeVuelo* (o *Especificaciónde-Vuelo*) que describa un vuelo y su ruta, aun cuando no esté programado un vuelo determinado (figura 9.9).

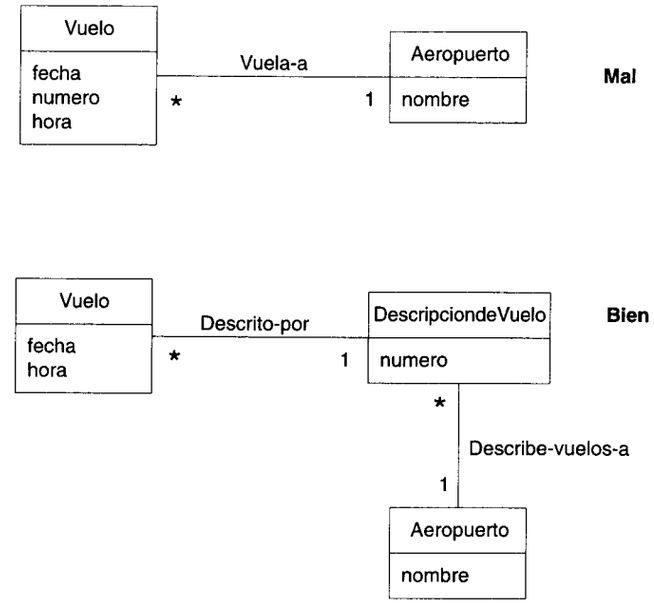


Figura 9.9 Especificaciones de otros elementos.

9.10 Definición de términos en el lenguaje UML

En UML, se emplean los términos "clase" y "tipo", no así "concepto". No existe consenso unánime respecto al significado de clase y tipo; así que para evitar ambigüedades el UML define rigurosamente ambos términos tal como se utilizan en su metamodelo (el modelo de UML), aunque otros autores y profesionales usan definiciones alternas y antagónicas.

Cualquiera que sea la definición, lo importante es su utilidad para distinguir entre la perspectiva de un analista de dominios que observa conceptos reales, como venta, y los diseñadores de software que especifican entidades de programas; por ejemplo, la clase *Venta* en Java. El UML sirve, entre otras cosas, para explicar concretamente las perspectivas de notación y terminología muy afines; de ahí la importancia de tener presente qué perspectiva va a adoptarse (un análisis, un diseño o una vista de la implementación).

> Para simplificar nuestra exposición, en este libro con el término "concepto" designaremos cosas del mundo real y con el de "clase", las especificaciones e implementaciones del software.

La definición de **clase** en UML es "una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica" [BJR97]. Algunos autores limitan la definición de clase a una implementación concreta de software, digamos una clase en Java [Fowler96]. Pero, en el UML, este vocablo tiene una acepción más general: abarca especificaciones que anteceden a la implementación. En ese lenguaje, a una clase implementada de software se le llama más concretamente **clase de implementación**.

En UML, una **operación** es "un servicio que puede solicitarse a un objeto para que realice un comportamiento" [BJR97], y **método** es la implementación de una operación que especifica el algoritmo o procedimiento de esta última.

La definición de **tipo** en UML se asemeja a la de clase —describe un conjunto de objetos parecidos con atributos y operaciones—, pero no puede incluir métodos. De ello se deduce que un tipo es una *especificación* de una entidad de software y no una implementación. Ello significa también que un tipo de UML es independiente del lenguaje.

Aunque no sea estrictamente exacto dentro del contexto del lenguaje UML, este libro a veces utiliza los vocablos "concepto" y "tipo" indistintamente, porque en el uso coloquial el vocablo "tipo" a menudo se define como sinónimo de un concepto del mundo real [MO95].

El término **interfaz** se define como un conjunto de operaciones visibles en el exterior. En el UML, puede estar asociada a tipos y clases (y también a paquetes que agrupan elementos). Aunque los conceptos reales pueden tener una interfaz (la de un teléfono, por ejemplo), el término suele emplearse dentro del contexto de una interfaz para entidades de software, como la interfaz *Runnable* de Java.

9.11 Modelos Patrón

El Modelo Conceptual se compone de diagramas estáticos de estructura UML que ilustran conceptos en el dominio, como se muestra en la figura 9.10.

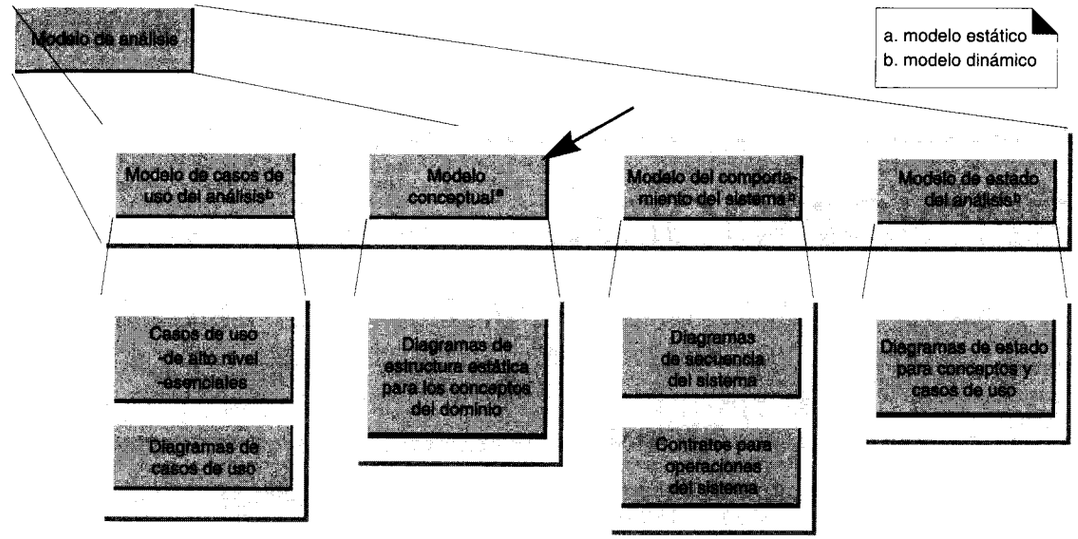


Figura 9.10 El modelo de análisis.