Modelado de dominio

Tabla de contenido

Introducción

Modelos

Ya hemos visto la importancia que tiene el modelado en las actividades de ingeniería de requerimientos, hemos visto que un modelo es una simplificación de la realidad. El modelo no es la realidad, sino que nos da una perspectiva de lo que queremos construir o de lo que hemos construido. Los sistemas que nosotros desarrollamos en general son complejos y necesitamos utilizar los modelos para poder visualizarlos, para poder especificarlos, para guiar la construcción, para de alguna manera documentar las decisiones que tomamos.

Modelado del dominio

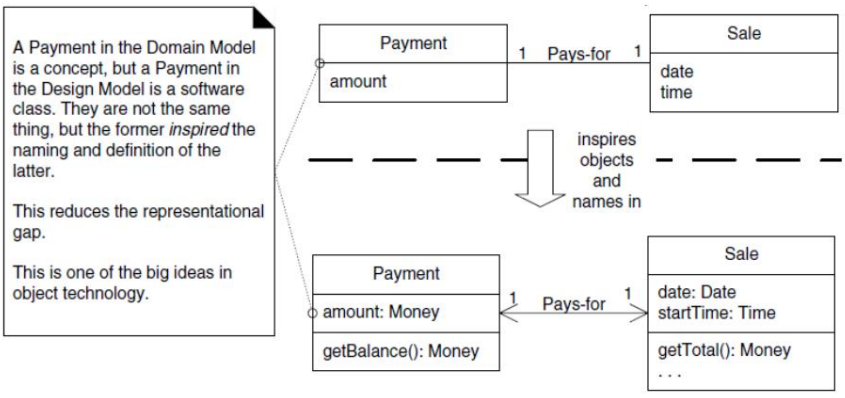
Vimos que hay distintos tipos de modelos, algunos más relacionados con el análisis de los requerimientos, con el modelado del problema, otros más orientados a un modelado de la solución, al diseño. En particular estamos hablando hoy del modelo de dominio. Un **modelo de dominio** es una representación visual del vocabulario del problema, del vocabulario del dominio del problema. Los objetos del modelo de dominio tienen que ver con ese vocabulario que emplea el usuario para contarnos cuál es el problema que tiene, para describirnos como es su operatoria, para describirnos qué es lo que hace. Si estamos pensando en un sistema bancario, probablemente los objetos de dominio o clases conceptuales más importantes sean el cliente, la cuenta corriente, la caja de ahorro, el movimiento, el saldo. Si pensamos en función del sistema de inscripciones de una facultad, los objetos de dominio más importantes van a ser las materias, los cursos, los alumnos, la inscripción. Construimos este tipo de modelos para entender mejor, y analizar mejor, el contexto en el que va a operar el sistema que tenemos que desarrollar.

En general, vamos a encontrarnos con que los objetos de dominio pertenecen a alguna de las siguientes grandes familias:

- **Cosas** manipuladas en una organización: Qué maneja la organización. Por ejemplo: contratos, facturas, pedidos.
- **Objetos** y conceptos del mundo real que el sistema necesita conocer y/o monitorear. Por ejemplo: un avión, una trayectoria, un misil.
- **Eventos** pasados o futuros. Por ejemplo: Un pago de una factura, la programación de una función de cine.
- **Personas** o **roles** de esas personas en organización o en el mundo del dominio, **u organizaciones**, por ejemplo, el cliente, el socio, el alumno.

[**V** / F] Un modelo de dominio es una representación visual del vocabulario del dominio del problema.

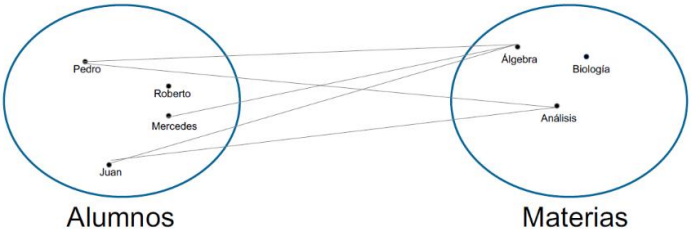
Lo que hay que tomar en cuenta es que estos objetos de dominio no son los objetos de diseño. Eso es muy importante. Nosotros podríamos decir que un cliente se caracteriza por tener un CUIT, un nombre, una dirección y no va a faltar que un día, bueno, pero un nombre es de clase string, dirección también… esas son clases que tiene que ver más con la implementación. Cuando hablamos de objetos de diseño estamos hablando de los grandes elementos que forman parte del vocabulario: El cliente, la factura, el ítem de la factura, la transacción, la operación, no son objetos de diseño. Por supuesto que uno de los conceptos fundamentales de la orientación a objetos, es minimizar la brecha entre el vocabulario del problema y la solución, pero, a no confundirse, no todo objeto de diseño existe en el dominio del problema. Hay que tener mucho, mucho cuidado, con esto.

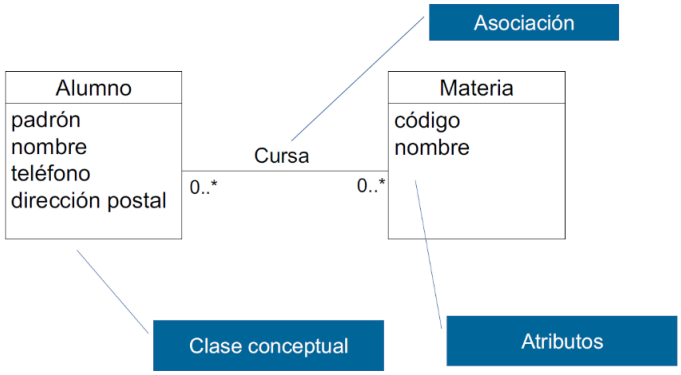


Modelado del dominio: objetos, asociaciones y atribuciones

El modelo de dominio está profundamente basado en teoría de conjuntos.

Aquí tenemos un conjunto de alumnos, un conjunto de materias, y vemos que hay instancias en ambos conjuntos. Que a su vez se asocian unos con otros, por ejemplo, Pedro está cursando álgebra y análisis; Roberto no está cursando ninguna materia. Podríamos presentar esto a través de un diagrama que utiliza notación UML, y vamos a ver que tenemos una clase alumno, una clase materia que están representando objetos del dominio. Vemos que esas 2 clases conceptuales se relacionan entre sí mediante una asociación, en este caso alumno cursa materia (la asociación es cursa), y vemos también que cada uno de estos objetos, cada una de estas clases, tienen atributos. Esos atributos son propiedades que caracterizan a cada uno de esos objetos. Esto va a formar la base, como decíamos antes, del diseño. Seguramente en nuestro diseño vamos a tener un objeto alumno (vamos a tener una clase alumno para ser más preciso), una clase de materia, pero fíjense que acá no estamos hablando de aspectos de implementación, no estamos hablando de métodos ni de operación, estamos hablando de clases conceptuales, objetos del dominio y propiedades o atributos.





[V / **F**] Todo objeto del diseño existe también como entidad del modelo de dominio del problema.

Modelo del dominio

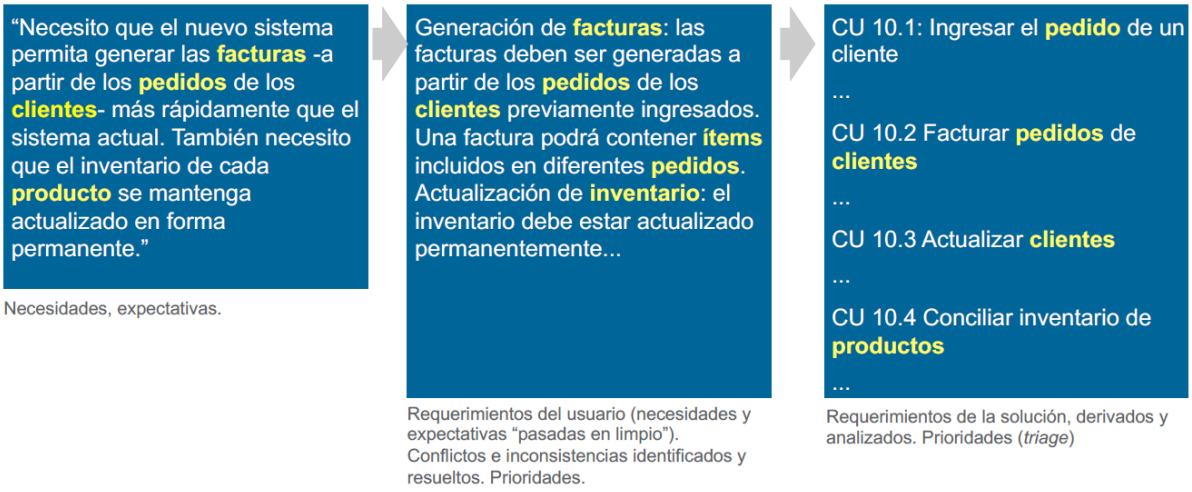
¿Cómo se construye?

¿Cómo hacemos para construir un modelo de dominio? Bueno, no hay una receta, pero en general podríamos decir que el primer paso es identificar las clases conceptuales utilizando un par de técnicas que vamos a mencionar: Se puede dibujar, hacer un diagrama, utilizando la notación de UML con la salvedad de que no son objetos de diseño, no son objetos de implementación, sino que son abstracciones. Podemos agregar detalles tales como asociaciones, atributos, generalizaciones y especializaciones, composiciones, agregaciones. En general este es un proceso iterativo incremental. Puede ser el modelo de dominio algo muy informal, dibujado en una pizarra; puede ser algo más formal dibujado a través de alguna herramienta. Lo importante es, no tanto es el resultado, sino pensar y entender bien cuáles son los grandes elementos, cuáles son los grandes objetos que forman parte o los objetos más relevantes más que grandes, los objetos relevantes del dominio y cómo se asocian entre sí. ¿Por qué? Porque esa es la información que el sistema va a tener que manejar independientemente de cómo lo implementemos. El sistema va a tener que recordar que el alumno Juan Pérez se anotó para cursar la materia de análisis matemático. El sistema va a tener que recordar que Juan aprobó análisis matemático, en tal fecha con tal nota, entonces eso lo tenemos que de alguna manera, representar en el modelo de dominio.

¿Cómo encontrar clases conceptuales?

Hay 3 estrategias básicas para encontrar clases conceptuales: Una es aplicar el análisis lingüístico, la otra es utilizar algún tipo de listas tramada de categoría y otra es reutilizar algún modelo ya existente.

Análisis lingüístico



El análisis lingüístico es bastante simple de explicar. Buscamos en nuestras minutas de reunión, en nuestras especificaciones, en nuestros casos de uso, buscamos sustantivos o frases que incluyan sustantivos para identificar candidatos a posibles objetos. Por ejemplo, acá tenemos facturas, pedidos, clientes, productos. Hay que tener cuidado porque también van a aparecer atributos o propiedades de otros objetos (que también son sustantivos), con lo cual hay que andar con un poquito de cuidado.

Lo que hay que tener, sí, mucho, mucho cuidado, es con aplicar este enfoque de manera mecánica. Si todos los sustantivos que están en la especificación van a ser objetos, seguramente nos vamos a encontrar con que va a haber un objeto que es dirección del cliente, o dirección, o código postal, que son sustantivos pero no son objetos, en este caso. Con lo cual, ahí hay que tener cuidado para justamente evitar esta tentación de darle un grado de granularidad muy, muy, alto y caer en la tentación de meter temas que tienen que ver más con la implementación y no con el aspecto más conceptual de lo que estamos tratando de definir. Con lo cual es conveniente aplicar esta técnica combinada con la siguiente que es la lista de categorías.

Lista de categorías

Categoría	Ejemplos
Transacciones	Venta, reservación, pago.
Ítems de una transacción	Ítem vendido, ítem facturado
Producto o servicio relacionado con una transacción o con un ítem de una transacción	Artículo, vuelo, butaca.
Dónde se registra la transacción	Factura, asiento contable, manifiesto de carga.
Roles u organizaciones relacionadas con una transacción	Cajero, sucursal, cliente.
Lugar de la transacción o de la provisión del servicio.	Sucursal, aeropuerto, vuelo.
Eventos	Venta, pago.
Objetos físicos	Avión, automóvil, cronómetro.

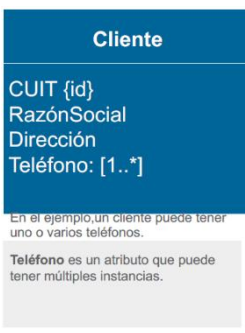
Descripción de cosas	Especificación de producto, descripción del vuelo.
Catálogos	Catálogo de productos, lista de precios.
Contendedores de cosas	Depósito, almacén.
Cosas en el contenedor	Ítems, pasajeros.
Sistemas externos que colaboran	Sistema de autorización, control de tráfico.
Registros financieros o de trabajos realizados, contratos, etc.	Recibo, bitácora de cambios.
Instrumentos financieros	Efectivo, tarjeta de crédito.
Cronogramas, manuales, etc.	Calendario de reparación, cronograma de proyecto.

En la lista de categorías, aquí lo que tenemos son algunas categorías de tipos de objetos que nos pueden orientar en la búsqueda de objetos. Por ejemplo, transacciones, ítems de una transacción, productos o servicios, los lugares en donde se registran las transacciones. Pensemos, por ejemplo, cuando vamos a un comercio a comprar algún tipo de artículo y nos emiten una factura. Pensemos ahí, cuáles son los objetos que están relacionados: Estoy yo como cliente, está la empresa probablemente como proveedora (tema que debemos discutir después), está el objeto que esté comprando (el producto que estoy comprando). El producto tendrá sus características: tendrá un precio, tendrá una descripción. También está la transacción en sí misma que queda reflejada a través de una factura, por ejemplo, que va a tener una serie de atributos que tendremos que analizar. O, por ejemplo, hacemos una reserva para ir a ver una película. Fíjense los objetos de dominio que podrían aparecer ahí: Nosotros como clientes, la película que vamos a ver, en qué sala la vamos a ver, la transacción propiamente de compra que nos va a indicar: compraste una entrada para tal películas para tal fecha en tal asiento, etc. Así que aquí tenemos una lista de categorías también, por ejemplo, roles, lugares en donde se provee servicio o tiene lugar la transacción; objetos físicos, a veces necesitamos tener información de objetos físicos. Por ejemplo, en algunos casos es importante que identifiquemos cada uno de los productos que hemos fabricado, por ejemplo, tal auto con tal número de serie, con tal número de chasis.

[V / **F**] Al momento de identificar las clases conceptuales utilizando el análisis lingüístico generalmente todos los sustantivos se representan como objetos.

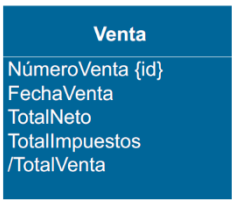
Atributos

Habíamos dicho que las clases conceptuales o modelos del dominio tienen atributos. Un atributo es un dato lógico acerca de un objeto. Por ejemplo, acerca de un cliente, nos interesará conocer su clave única de identificación tributaria o CUIT, su razón social, su dirección, su teléfono. En este caso teléfono, es un atributo repetitivo, es decir, tenemos varias instancias de ese atributo. Lo que no tenemos que caer nuevamente en la tentación, es decir que el CUIT es una clase en sí misma, la razón social es una clase que pertenece a tipo string, etc. Estamos hablando de vuelta de, en este caso, datos primitivos. Si es que nos interesa a esta altura decir que algo es numérico o no. Quizá en muchos casos va a valer la pena y en otros mejor no conviene hacer esa profundización.



En el comienzo del análisis orientado a objetos, cada instancia de una clase conceptual, existía por el medio hecho de existir, y no necesitaba ningún tipo de identificador para diferenciar una instancia de la otra porque se asumía que cada instancia de la otra se podía autoidentificar de alguna forma que no era relevante para el análisis del problema. Pero de todas maneras, hay casos en donde hay identificadores naturales, por ejemplo, la clave única de identificación tributaria o CUIT es un identificador natural. Entonces, podemos poner un estereotipo como el que figura ahí entre llaves y "*id*" para indicar que es un identificador natural. El DNI es un indicador natural, el número de padrón es un identificador natural, con lo cual conviene indicarlo ahí. Ya en tiempo de implementación, cuando transformamos esto, por ejemplo, en una tabla de clientes, la clave primaria será el CUIT o el

DNI, o el número de padrón, pero eso es algo de implementación, no es un problema que nos preocupa acá, pero podemos, de todas maneras, si ir identificando que hay determinados atributos que nos permiten individualizar cada una de las instancias de un determinado objeto de valor.



También es importante que algunas veces mostremos atributos que son derivados a partir de otros. En el ejemplo, TotalVenta es un atributo derivado que resulta de la suma entre TotalNeto y TotalImpuestos. Esto lo podemos poner siempre y cuando sea relevante para entender el dominio. No tenemos que poner atributos ahí que tengan que ver más con un tema de implementación. Digamos, si yo pongo TotalVenta porque digo "no porque desde un punto de vista de implementación va a ser más rápido, en lugar de tener que hacer el cálculo", o puedo decir lo contrario "no, no pongo TotalVenta que es un atributo digamos significativo desde el punto de vista del dominio porque me va a ocupar mucho espacio en la base de datos", no, es ridículo ese planteo porque acá de lo que estamos hablando del dominio del problema. Con lo cual restricciones de implementación acá no aplica. Estamos en un nivel de abstracción muy alto como para ponernos a discutir de esos temas.

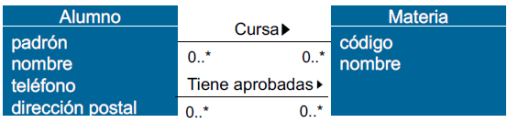
[V / **F**] Un atributo referenciado como "Teléfono: [1..*]" indica que existe una relación con la entidad Teléfono de entre 1 y muchos.

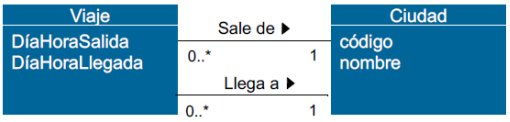
Asociaciones

El otro tema sumamente importante son las asociaciones. Las asociaciones representan una relación entre instancias. Al igual que los atributos, que tienen que ser recordados por el sistema, las asociaciones también tienen que ser recordadas por el sistema. Cada asociación tiene un nombre y tiene una multiplicidad. En el ejemplo, un alumno puede estar anotado en cero o en muchas materias, y en una materia puede haber anotados muchos o ningún alumno. Por ejemplo, si la materia es nueva, todavía no va a tener alumnos inscriptos. Lo que hay que tomar en cuenta es que la multiplicidad nos está indicando una regla de negocio. En este caso, la regla de negocios es que un alumno puede cursar cero, una o muchas materias y que puede haber casos en donde las materias no tengan alumnos porque todavía no se están dictando o porque todavía no es el momento de inscribirse o porque no se anotó nadie para cursarla. Hay que tener cuidado, una cosa es una asociación y otra cosa es un atributo. No tenemos que usar, reitero, no tenemos que usar atributos que referencien a otros objetos. Yo podría decir ahí que, por ejemplo, ¿podría tener algún tipo de puntero desde materia a los alumnos inscritos y que eso fuera un atributo que estuviera incluido en materia? No. Eso es un tema de implementación. ¿Ok? Sí, cuando tengamos que implementar esto en una base de datos, probablemente tengamos una tabla de alumnos, una tabla de materias y tengamos una tabla de correlación que implemente la relación entre alumnos y materia, pero, ojo con eso, acá estamos hablando de asociaciones que son conceptuales que no implican ningún tipo de atributo referencial. No tenemos que poner atributos que referencien a otros objetos, porque en realidad, para eso, para mostrar eso, tenemos las asociaciones.

Asociaciones binarias

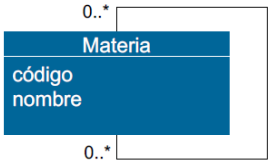
Las **asociaciones** típicas son **binarias** entre 2 objetos. Puede haber más de una asociación entre un par de objetos. Tranquilamente en este caso, un alumno puede cursar una materia y puede tener aprobadas otras materias. En viaje siempre tenemos una ciudad de destino y una ciudad de llegada. Eso es perfectamente normal. Recuerden que las asociaciones relacionan distancias de las clases conceptuales que forman parte de la relación.

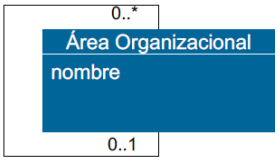




Asociaciones unarias

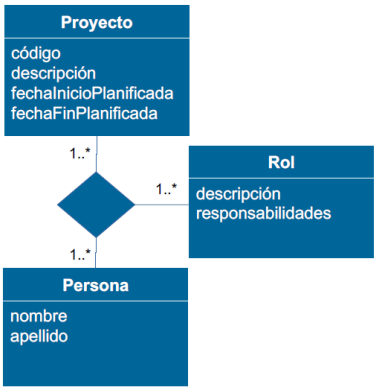
Típicamente las asociaciones son binarias, pero también puede haber **asociaciones unarias**. Por ejemplo, una materia puede tener como correlativas otras materias y una materia es correlativa de otra. Lo que hay que tener en cuenta ahí, en este tipo de relaciones jerárquicas, es que no todas las materias tienen materias que les sigan, ni todas las materias tienen materias anteriores (las materias del CBC no tienen materias anteriores). Para cursar introducción al conocimiento científico probablemente no necesiten tener una materia aprobada desde antes. Siempre que uno representa organizaciones o estructuras jerárquicas hay que tener cuidado con los nodos y las hojas. Por ejemplo, área organizacional puede tener, ahí lo que representa, imagínese un organigrama. Entonces, la raíz del organigrama: del gerente general dependen las gerencias; de las gerencias depende las jefaturas; de las jefaturas dependen los empleados. Ahora hay empleados que no tienen ningún tipo de estructura jerárquica por debajo, con lo cual ahí, el árbol se termina, con lo cual hay que tener cuidado con la cardinalidad, por eso está el cero a muchos, el cero a uno. Hay que mirarlo con detalle.





Asociaciones ternarias

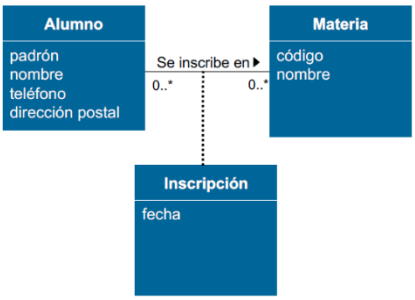
Si bien son extrañas, pueden aparecer **asociaciones** de grado mayor a 2. Las más comunes son las **ternarias** puede haber de grado 4, 5, pero ya son más raras de ver. En este ejemplo en particular, lo que estamos viendo es que una persona puede tener más de un rol en un proyecto determinado. Eso es lo que está representando esa asociación. Recuerden que en este caso, cada instancia de la asociación está representando una relación entre persona, rol y proyecto.



En el ejemplo, una persona puede tener más de un rol en un proyecto determinado.

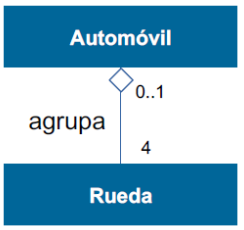
Clases asociativas

En algunos casos nos vamos a encontrar con unas asociaciones medio extrañas que nos vamos a dar cuenta que, además, tienen atributos que son propios. En ese caso no vamos a estar ante una asociación, sino que vamos a estar ante lo que se llama un **tipo de objeto asociativo o clase asociativa**. La clase asociativo tiene la particularidad de *comportarse como una clase y como una asociación al mismo tiempo*. Entonces, en este ejemplo, lo que vemos es que un alumno se inscribe en una materia. ¿Cuándo se inscribe? En una fecha determinada. Entonces ahí lo que necesitamos es un objeto asociativo "inscripción" con el dato "fecha". No podemos tener fecha como un atributo de una asociación porque eso, conceptualmente, no es viable.

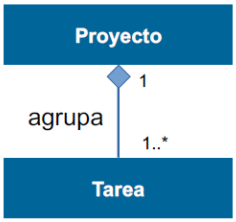


Composición y agregación

Hay un tipo particular de asociaciones que son las que permiten representar las relaciones de todo y parte. Por ejemplo, un automóvil tiene ruedas. En este ejemplo, tiene cuatro. Pero las ruedas son también elementos que son independientes (las puedo sacar).



En cambio, en un proyecto, yo tengo varias tareas; pero esas tareas, si las saco del proyecto, no existen, no tiene existencia por la independencia. Entonces, lo que propone UML es este tipo de asociaciones que llaman **composición y agregación**. *Cuando tenemos una colección de elementos que no depende del ciclo de vida del contenedor* (el caso del automóvil y la rueda), *estamos ante una agregación y se representa con ese diamante sin rellenar*. Ahora, *cuando los elementos de la colección dependen del contenedor* (tarea con respecto a proyecto), *estamos ante una composición*. ¿Cuándo es conveniente utilizar este tipo de representación o de asociación? ¿Cuándo es conveniente utilizar una asociación común? Porque esto mismo que estamos viendo acá, se podría representar utilizando una asociación de las que vimos antes. La verdad es que el criterio que debería primar es el de la claridad en la representación. Digamos, si todos entienden de lo que estamos hablando, probablemente convenga utilizar este tipo de estructura.

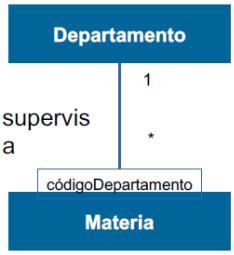


¿Cuáles de los siguientes son buenos nombres de asociaciones?

- ☑ **Registra**
- ☑ **Cursa**
- ☐ Tiene
- ☐ Utiliza
- ☑ **Es correlativa de**
- ☑ **Escribe**

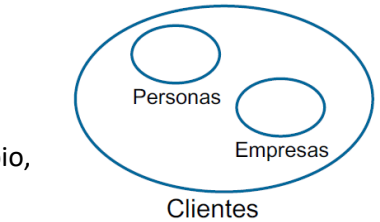
Asociaciones calificadas

Hay un tipo de **asociaciones** que se llaman **calificadas**. Supongamos por un momento que el departamento de computación aglutinara a todas sus materias en "75algo" y que el departamento de Computación sea el departamento 75. Entonces está la materia 7501, la materia 7509 (como la nuestra), etc. Entonces la materia queda identificada a partir de su propio código (09) más el código del departamento (75). Entonces, en este caso, hablamos de una asociación calificada *porque distinguimos un grupo de distancias en uno de los extremos mediante un valor calificador*, que en este caso es el código del departamento. Entonces, si hiciéramos doble click en "materia", nos encontraríamos que materia tiene como atributos código de materia y el nombre. Pero para poder identificarse necesita del código del departamento al que pertenece, porque, por ejemplo, podría haber una materia 09 en otro departamento. ¿Ok? Esto es algo muy, muy importante y muy útil para el modelado.



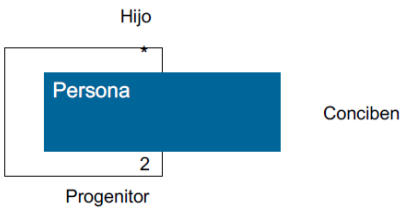
Generalización y especialización

También hay situaciones en donde vamos a necesitar hacer algo que es clásico de la orientación a objetos, que es establecer **generalizaciones y especializaciones**. En el ejemplo estamos viendo que para nosotros es importante el cliente, pero tenemos dos tipos de clientes, que tienen atributos propios. Por ejemplo, tenemos clientes que son personas físicas, de las cuales nos interesa su apellido, su nombre, su dirección. En cambio, hay otros clientes que son empresas, y de esos en particular, lo que nos interesa son la razón social, su dirección comercial, su dirección legal, su CUIT. Entonces probablemente necesitemos modelar esto como una clase padre, que tiene un atributo que si nos interesa, que es el código del cliente, y 2 clases hijas que heredan parte de la definición de cliente pero tienen sus propias particularidades. ¿Cuándo hacemos esto? Cuando realmente nos interesa mostrarlo porque necesitamos mostrar atributos que son diferentes, porque hay asociaciones distintas entre los hijos (por ejemplo: personas se puede asociar con determinado tipo de objetos y empresa con otros). De vuelta, el criterio que tiene que considerarse aquí es claridad en la representación, claridad en la comunicación es lo más importante.



Roles en una asociación

También puede haber **roles en una relación**. Eso *ayuda a clarificar el papel que juega cada uno de los objetos en la relación*. En este caso, vemos que una persona es concebida como resultado de la interacción de 2 personas. En un extremo tenemos el rol de una persona que juega el rol de hijo, y en el otro extremo, la persona juega el rol de progenitor (como nos pasa a todos). No es obligatorio incluir el tema de los roles, pero sí ayuda a mejorar la relación entre objetos.



¿Cómo encontrar asociaciones?

Nuevamente para encontrar asociaciones podemos utilizar categorías:

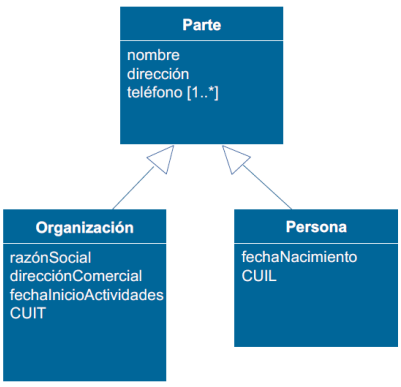
Categoría	Ejemplos
Una transacción relacionada con otra.	Venta (o factura) - Pago.
Un ítem de una transacción.	Ítem factura - Factura.
Un producto o servicio incluido en una transacción (o en un ítem)	Vuelo – Reserva; Compra – Ítem Comprado
Un elemento forma parte de otro (física o lógicamente)	Sala – Butaca Computadora – Procesador
Roles en una transacción.	Cliente (en una compra o factura) Comercio (Cupón Tarjeta Crédito)
Descripción de algo.	Especificación (de un producto) Descripción de vuelo.
Registración, conocimiento.	Reserva (de una habitación)
Algo que es miembro de otra cosa.	Cajas (en un supermercado). Socios (de un club). Empleados (de una organización)
Una subunidad organizacional.	Departamento (dentro de una gerencia)
Algo administrado o usado.	Chofer (conduce un micro) Complejo de cines (administra salas)

Patrones

Vamos a revisar rápidamente algunos patrones que habitualmente nos vamos a encontrar. Patrones que tienen que ver con el modelado de dominio. Vamos a repasar algunos de estos:

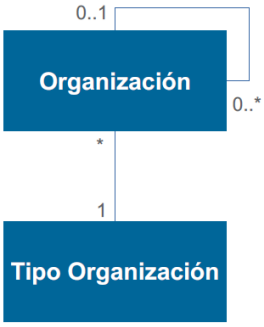
Personas y organizaciones

Algo que es típico es el de personas y organización. Medio parecido a lo que veíamos hace un minuto. Si agarramos la agenda de nuestro teléfono, vamos a encontrar con que hay personas pero también hay organizaciones. Entonces… ¿cómo hacemos? ¿No hay atributos comunes entre ellos? ¿No hay atributos particulares para la persona y atributos para la organización? Entonces algunos autores proponen un patrón de modelado que lo llaman "**parte**" (sea la traducción de party en el libro de Fowler). Sinónimos: Persona física/jurídica, tercero, etc. Fíjense que acá el objeto de dominio padre "parte" tiene nombre, dirección, teléfono, atributos que son comunes a una persona o a una organización. En particular, organización tiene razón social, dirección comercial, fecha de inicio de actividades, CUIT. Mientras que persona, tiene fecha de nacimiento, CUIL. Nuevamente, es sólo un ejemplo, puede haber más o menos atributos y a lo mejor puede pasar que no nos interese realmente modelar esto de esta manera porque no es relevante para el dominio.

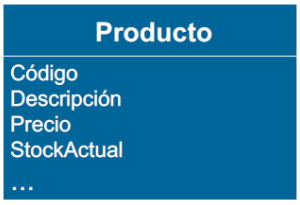


Es sólo un ejemplo. Puede haber más o menos atributos, dependiendo de las situaciones particulares.

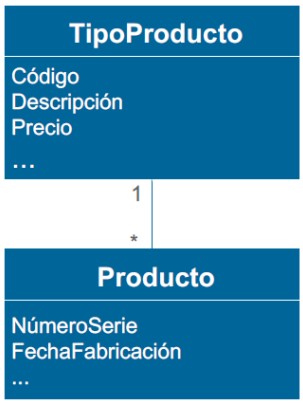
El tema de las **estructuras jerárquicas** también es un tema importante. Podríamos modelar la estructura de una empresa con este modelo de dominio que estamos viendo ahí (que es espantoso) o podríamos hacer algo un poco más inteligente, y utilizar este tipo de estructuras, en donde hay una unidad organizacional, que puede tener cero, una o muchas unidades organizacionales que las reporten. Y podríamos decir que cada unidad organizacional pertenece a un tipo de unidad organizacional determinada. Por ejemplo: departamento, gerencia, dirección. Nuevamente, fíjense que hay unidades organizacionales que son hojas en un árbol y no tienen unidades organizacionales hijas y hay nodos en el organigrama (la gerencia general) no tiene alguien que esté por arriba. Con lo cual, por eso es importante entender la cardinalidad de la asociación: 0…1, 0…*(muchos), etcétera. La pregunta que les dejo para que analicen es: ¿soporta este modelo una estructura matricial?, ¿soporta este modelo posibles cambios a la estructura organizacional? Por ejemplo, que se fusionen las gerencias o que desaparezcan las gerencias y desaparezca un nivel en el organigrama.



Productos y especificaciones



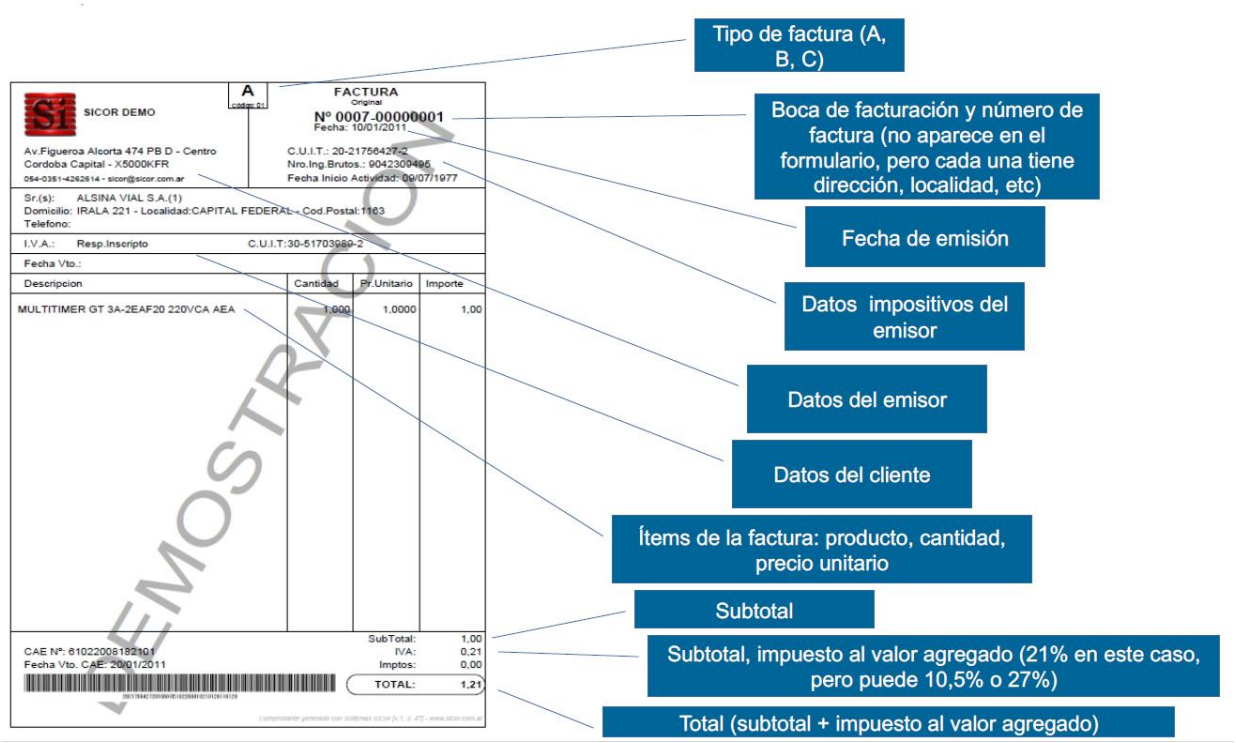
Una de las cosas que los sistemas, en general, siempre necesitan conocer es qú tipo de productos se fabrican en la organización, qué stock hay de esos productos. Y acá hay 2 alternativas importantes: A veces no nos interesa identificar cada unidad de los productos que fabricamos. Por ejemplo, si producimos frascos de mermelada probablemente nos interese saber que tenemos en stock 20 unidades de mermelada de naranja. A lo mejor no nos interesa obtener información de CADA frasco de mermelada de naranja. Si fabricamos autos, probablemente nos interese saber el número de serie de cada auto. Entonces acá, hay que analizar muy cuidadosamente si nos interesa tener información de cada producto fabricado en forma individual. Entonces, en esos casos, en los casos que sí nos interese manejar cada uno de los productos en forma individual, probablemente tengamos que tener un tipo de producto que nos describa las cosas comunes. Por ejemplo: Mermelada de naranja que se vende a $ 60 el frasco; de las estancias de esos frascos de naranja, por ejemplo, hay un frasco de naranja, número de serie: 300.223 que está en tal negocio, pertenece a tal lote de producción, que se fabricó tal día y salió a tal hora de la fábrica. Depende nuevamente, del dominio del problema que estemos analizando.



Facturas

La factura es un objeto que uno lo ve a simple vista y parece sencillo: Información de quién es el cliente, de quién es el que está emitiendo la factura, hay un número de factura, hay un detalle de lo que estoy comprando, etc. La factura es un documento que refleja la información de una operación de venta (es la que nos dan cuando vamos a cualquier comercio). Cada país tiene regulaciones muy particulares. En Argentina, ustedes han visto que las facturas pueden ser A, B o C dependiendo de distintas particularidades. No solamente hay que mantener la copia impresa, sino que también hay que mantener la información electrónica de esa factura. Actualmente se están imprimiendo lo que llamamos la factura electrónica, con lo cual uno genera un archivo PDF con la imagen.

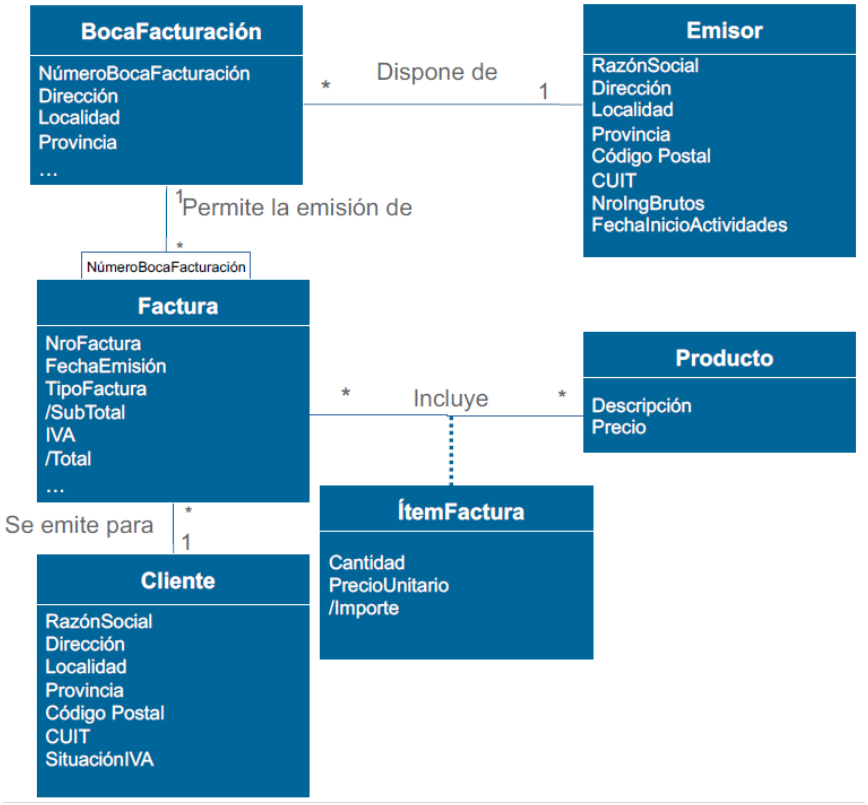
Ahora, qué información tenemos acá:



Vamos a encontrarnos con que hay un tipo de factura (A, B, C), que depende de si la factura que estoy emitiendo es para alguien que necesita que le discrimine el IVA o no. Después hay un número de factura que incluye la boca de facturación (que son esos primeros cuatro dígitos más un número correlativo). Por ejemplo: ¿qué es una boca de facturación? Y… yo puedo tener distintas sucursales. Entonces la sucursal número uno o la sucursal de Barracas tendrá la boca de facturación 0001 y emitirá facturas que van a empezar con el número 0001 y va a seguir después con 7 u 8 números secuenciales que tienen la información del número de factura propiamente dicho. O, en un mismo lugar, puedo tener más de una boca de facturación. Quiero decir, puede tener más de un "talonario" la factura. También hay información con respecto a la fecha de emisión de la factura, los datos impositivos de quien está emitiendo la factura (del vendedor), los datos del cliente (dirección, teléfono), la condición frente al IVA del cliente/destinatario de la factura. Después están los ítems de la factura, que es el detalle de los productos y servicios que estoy comprando, la cantidad, el precio unitario, el importe total. Al final vamos a tener un subtotal. Vamos a tener un subtotal con el IVA (que en algunos casos es el 21% sobre el precio neto o precio unitario, en algunos otros es el 10,5% en otros el 27%, eso varía). Y el total, que va a ser en definitiva el total neto (sin los supuestos) más el IVA. Esto es una recontra simplificación porque también podrían haber impuestos internos incluidos aquí, pero mantengamos el ejemplo sencillo. Entonces: ¿cómo hacemos?

Acá hay varias cosas: Si se fijan candidatos a objetos de dominio hay montones. Pensemos por un momento: Esta la factura propiamente dicha, pero la factura tal cual la tenemos representada acá, en realidad está asociada con otros objetos si yo tuviera que representar esto como un objeto de dominio: El cliente, el emisor, los ítems, etc.

Vamos a ensayar una posible representación de esto:

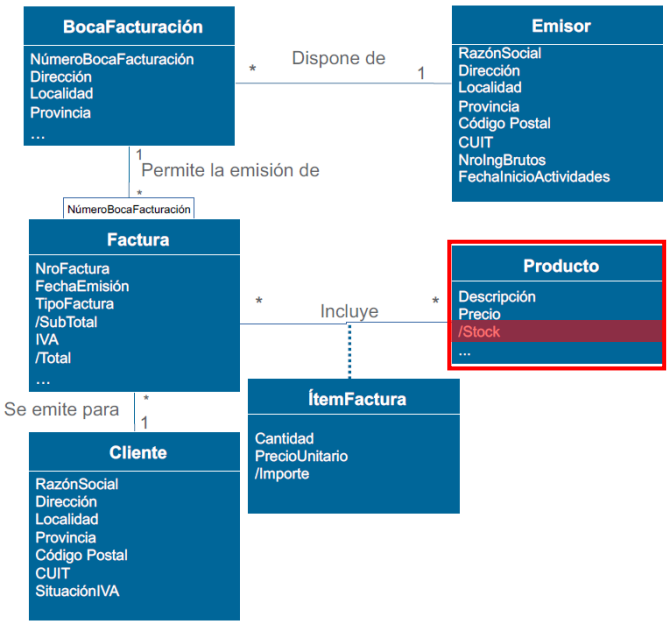


Ejemplo 1

Ahí tenemos un posible ejemplo. Tenemos el emisor que es el que emite la factura; dispone de una o varias (el * representa: 0, 1 o varias. A veces UML nos propone que pongamos explícitamente si es cero o muchos) bocas de facturación. Esa boca de facturación tiene una dirección, una localidad, una provincia. Después tenemos la factura, que en realidad la factura lo que estamos incluyendo acá son los datos que son comunes a las facturas: el número, la fecha de emisión. Fíjense que factura tiene una relación ahí calificada con respecto a boca de facturación por el tema de la identificación que habíamos comentado antes. Pero también lo que vemos es que cada factura incluye varios ítems. Entonces, lo que estamos viendo acá en esta relación, en esta asociación “incluye” es este detalle, que es el que normalmente tenemos acá: “itemFactura”: Cantidad, precio unitario, importe. Pero cada ítem referencia a un producto y obviamente está asociado con la factura. La factura es un objeto compuesto por varios ítems de factura asociados a un emisor, a una boca de facturación, a un cliente. La pregunta que yo les haría acá es por qué hay un precio unitario en el ítem de factura. ¿Por qué, por ejemplo, ese precio también está en el producto? Bueno, la explicación es sencilla: porque los precios varían y varían en el tiempo. Una cosa es el precio que tiene hoy el producto y otra cosa es el precio al cual yo lo vendí. Con lo cual, a veces parece que hay redundancia de información, pero a veces no. No es redundancia, sino que tiene que ver con la definición misma del dominio del problema.

Facturas: otros escenarios

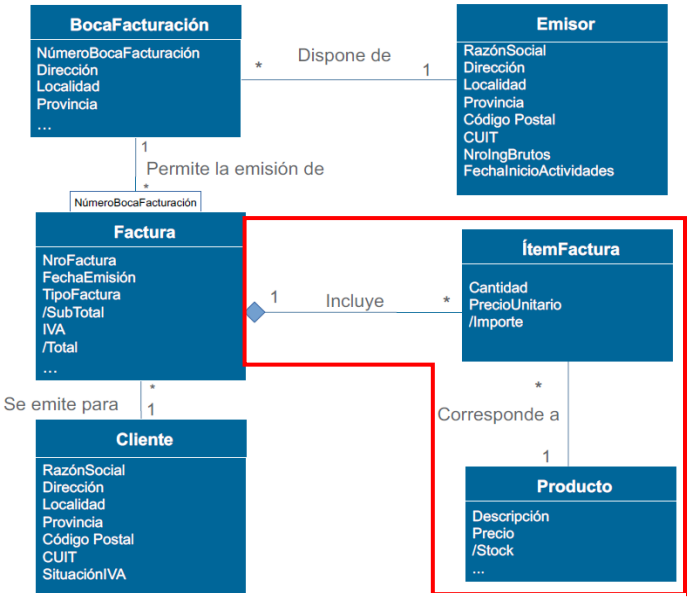
Preguntas: ¿Qué pasa si yo quiero repetir como ítem más de una vez el mismo producto? ¿Lo soporta el modelo? ¿Qué ocurriría si fuese necesario incluir el número de serie de un producto en particular? Acá (*Ejemplo 3*) me daría la sensación que producto ahí está representando un tipo de producto en realidad. La otra pregunta es si necesitamos siempre la clase cliente. A lo mejor podemos omitirla. ¿Harían falta las clases localidad y provincia? Posiblemente son atributos de cliente.



Ejemplo 2

Facturas: (algunos) otros escenarios

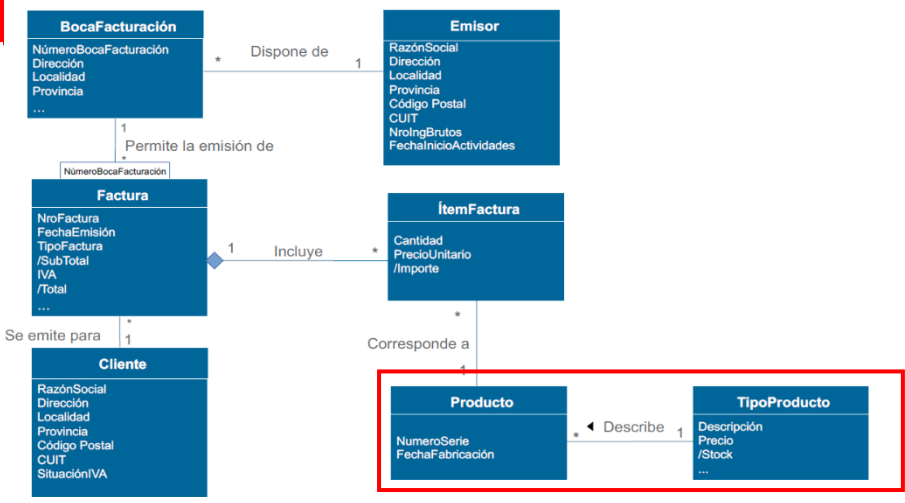
Vemos algunos otros escenarios:



Ejemplo 3

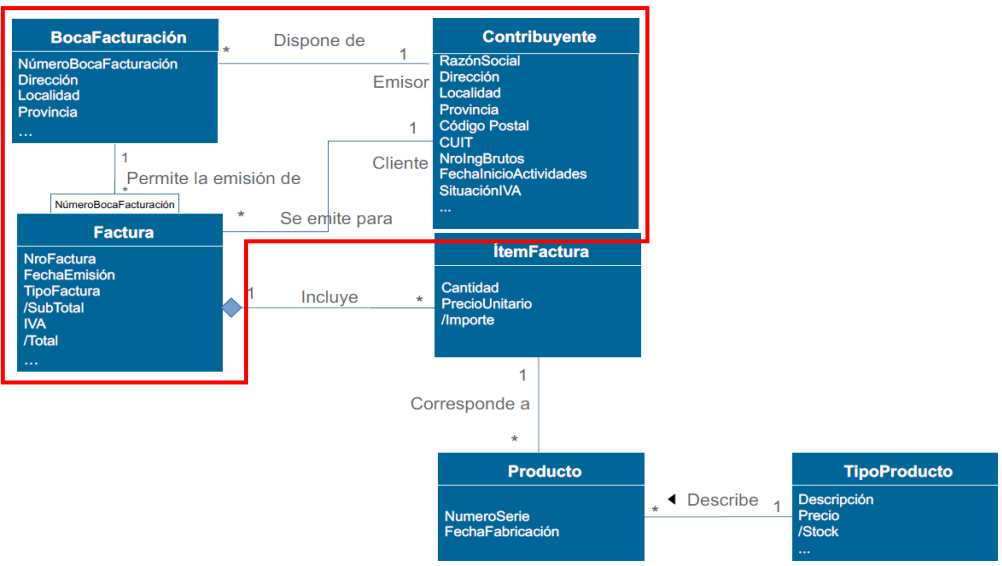
Acá (*Ejemplo 3*) lo que vemos es que facture incluye varios ítems de factura. Hay una relación de agregación. Y cada ítem de factura referencia a un producto. Con esto resolveríamos en parte, el tema de poder repetir el mismo tipo de producto más de una vez en la factura.

Acá (*Ejemplo 4*) modificamos el modelo para poder vender ítems o productos diferenciados, es decir, productos que necesitamos incluir el número de serie. Entonces, en este caso producto ya no representa una familia o un tipo de producto determinado, es el caso de que necesitamos vender en vez de la mermelada, el frasco de mermelada, número de serie 33.323



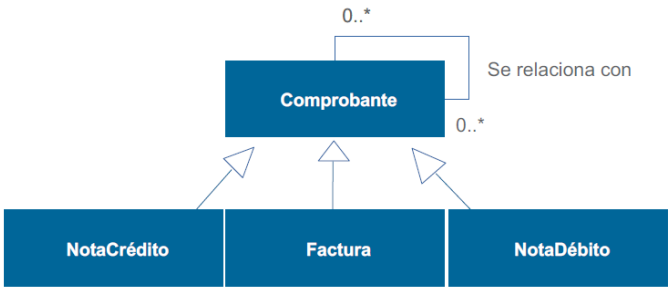
Ejemplo 4

La AFIP tiene un sitio web en el cual uno ingresa y genera facturas electrónicas. Y ahí, todos somos contribuyente. Nosotros (los que emitimos facturas) y los clientes, con lo cual, ya dejaría de existir el objeto cliente diferenciado del objeto emisor. Tendríamos un objeto contribuyente que puede cumplir el rol de emisor o de cliente dependiendo de la situación.



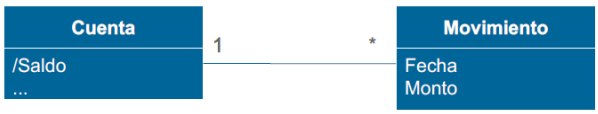
Ejemplo 5

Las facturas no están solas en la vida: hay notas de crédito, notas de débito. Las notas de crédito y notas débito son muy parecidas. Tienen un número, una boca de facturación, número de nota de crédito, de nota de débito. Pueden ser A, B o C. Representan cosas ligeramente diferentes: Las notas de crédito son saldos a favor del cliente, y las notas de débito son a favor del vendedor. Por ejemplo: Me devolviste un producto, yo te hago una nota de crédito por el monto del producto que me devolviste. Una nota de débito podría ser, me pagaste con retraso en la factura, te genero una nota de débito para que te quede a vos una deuda y a mí me quede esa deuda registrada. Esto ya es entrar en demasiado detalle, pero tranquilamente podríamos tener que representar eso de esta manera. Y comprobante se sucedía con emisor, con el cliente, podrían incluir distintos ítems, etc.



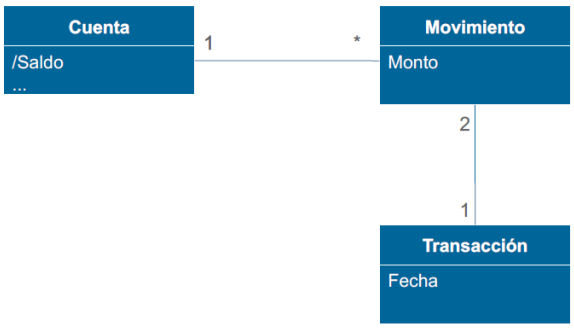
Inventario y contabilidad

Otro patrón interesante son los que tienen que ver con **inventario y contabilidad**. Muchas veces es necesario *registrar los movimientos y los saldos de bienes y valores*. Por ejemplo, cuando uno agarra la cuenta de la caja de ahorro o el saldo de la caja de ahorro uno se encuentra con que la cuenta tiene un saldo que es derivado de una serie de movimientos. Cuando uno ve su resumen de cuenta bancaria, se encuentra que para llegar al saldo actual en la fecha en la cual se emite el resumen hubo toda una serie de movimientos que sumaron y restaron valores y que nos permitieron llegar al número actual.



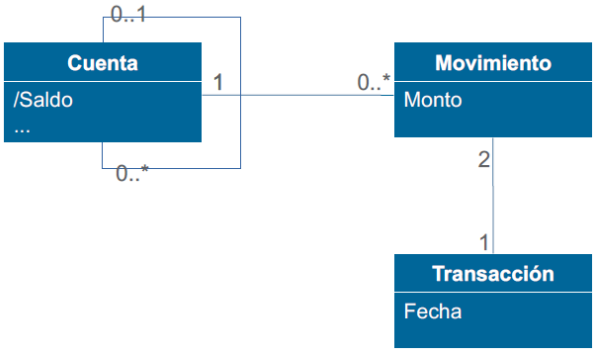
Lo mismo pasa con los productos. Hay movimientos de stock. Si yo necesito reflejar que hubo salidas de tanto frascos de mermelada, tendré que registrar esos movimientos de stock, en qué fecha fueron, qué cantidad. Eso se representa de la manera en la cual lo estamos viendo acá.

A veces también es necesario registrar el movimiento entre cuentas. Por ejemplo, transfiero plata de mi cuenta corriente a mi caja de ahorro o viceversa. Entonces eso en general queda representado por una transacción que tiene una fecha, podría también tener una identificación única; con 2 movimientos asociados, uno para debitar de la cuenta, es decir, para sacar por ejemplo, de la caja de ahorro $ 100, y otro para acreditar, en la otra cuenta, por ejemplo la cuenta corriente, es decir para sumar, $ 100. Eso se representaría de esta manera. Por supuesto, esto es una recontra simplificación, se puede complicar muchísimo más.



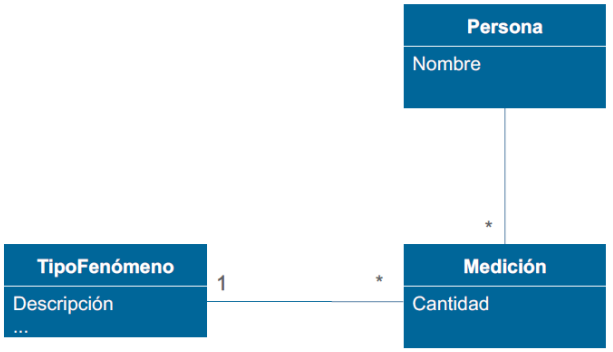
La vida real es mucho más complicada que este modelito, pero en lo conceptual, cuando necesitamos identificar cómo llegamos a un número a partir de una serie de movimientos, las estructuras que se utilizan, en general son estas.

Lo mismo con los esquemas contables. En la contabilidad hay movimientos de fondos entre cuentas de distintas jerarquías. Algunas de esas cuentas tienen subcuentas. Y ahí estamos, representándola, a través de esa estructura jerárquica. Y obviamente, saldo es un atributo derivado de los distintos movimientos que se hicieron sobre esa cuenta (o sobre las cuentas hijas). Probablemente haya cuentas de una determinada jerarquía que no tengan movimientos porque solamente suman cuentas contables que si tienen movimientos.



Mediciones

El tema de **mediciones** también es muy interesante. A veces necesitamos, por ejemplo, registrar que Roberto mide 1,72 M entonces lo que estamos representando en este modelo es que hay una persona que tiene un nombre, que tiene una medición que es una cantidad y que tiene que ver con un tipo de fenómeno que es la descripción, que en este caso es la altura. También podríamos refinar esto indicando la unidad de medición.



Ejemplos de patrones de modelado de dominio son

- ☑ **Organizaciones y personas**
- ☑ **Productos y especificaciones**
- ☑ **Mediciones**
- ☑ **Facturas**
- ☑ **Inventario y contabilidad**

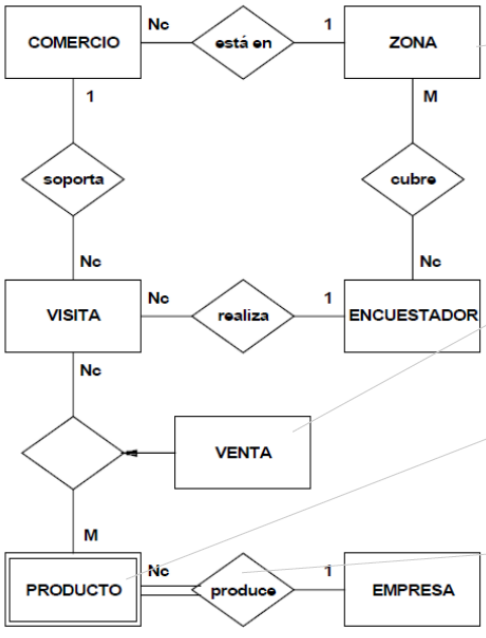
Modelado de datos

Métodos y modelos: una breve (e incompleta) perspectiva histórica

~1967: Programación Estructurada.
1970: A Relational Model of Data for Large Shared Data Banks (Codd)
~1973: SADT/IDEF0 (Ross)
~1975: Diseño Estructurado (Constantine & Yourdon) — Módulos, cohesión, acople
1976: Diagrama de Entidad-Relación (Chen) — **Modelado conceptual de datos**
~1978/9: Análisis Estructurado (Gane & Sarson, DeMarco) — Funciones, datos. Descomposición funcional.
1980: Smalltalk.
1981/3: Ingeniería de la Información (Finkelstein/Martin) — **Modelo de información**
1982: Diseño orientado a objetos [OOD] (Booch)
1984: *Essential Systems Analysis* (McMenamin & Palmer) — **Memoria esencial.**
1987: *A Framework for Information Systems Architecture* (Zachman) — **Arquitectura organizacional**
1988: *Object-Oriented System Analysis* (Schlaer/Mellor) — **Análisis orientado a objetos**
1989:*Modern Structured Analysis* (Yourdon)
1990/1: *Object Oriented Analysis; Object oriented Design* (Coad/Yourdon) — **Análisis orientado a objetos**
1991: Object-Oriented Modeling and Design (Rumbagh); *Object-Oriented Design With Applications* (Booch)
1992: *Object-Oriented Software Engineering* (Jacobson) — **Modelo de análisis, modelo de casos de uso**
1995: *Architectural Blueprints: The "4+1" View Model of Software Development* (Kruchten)
1997: UML v1.0 — Unificación de las notaciones propuestas por Booch, Rumbagh y Jacobson.
1998: Extreme Programing. — User stories
1999: Proceso Unificado. — Fases, iteraciones, modelos del sistema basados en "4+1"
2001: Agile Manifesto
2005: UML 2.0

El modelado de datos se inició prácticamente con el desarrollo de software. Hay como 2 grandes escuelas con respecto al tema de desarrollo de software. Por un lado, algunos especializados, mucho, en que lo más importante es la funcionalidad y las funciones, los datos vienen después. Y otros que decían, no, primero pensemos bien los datos y luego las funciones. Y hay infinitas combinaciones entre los 2 enfoques. El tema de modelado de actos arranca, probablemente, con el paper de Chen y su planteo del diagrama Entidad-Relación. Eso evolucionó con los años y yo diría que, la evolución que ha llegado a nuestros días es el modelado de dominio.

Diagrama de Entidad-Relación (ERD o DER)



Cada entidad y cada relación tiene una entrada en el diccionario de datos. Por ejemplo:
Comercio = @CUITComercio + RazonSocialComercio + TitularComercio + DomicilioComercio + TelefonoComercio
Zona = @IDZona + DescripcionZona + TotalHabitantesZona
Está en = {@CUITComercio} + @IDZona

Las entidades asociativas se comportan también como relaciones. Además de los atributos de la relación, tiene atributos propios:

Venta = @NroVisita + @IDEmpresa + @CodProducto + CantidadVendida

Las entidades débiles (o atributivas) tienen una dependencia de existencia con otra entidad (su identificación es una concatenación de el identificador del tipo de objeto fuerte con el que está vinculada y su propio identificador).

Producto = @IDEmpresa + @CodProducto + DescripcionProducto + PresentacionProducto

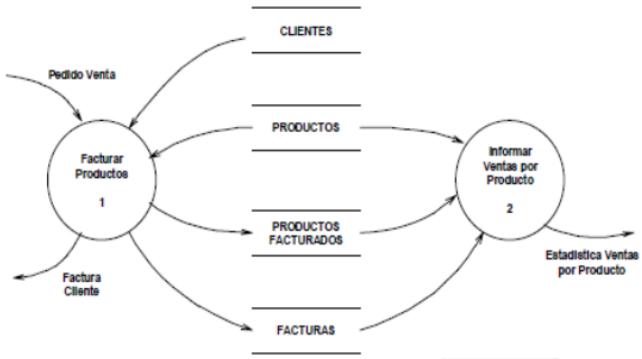
Las relaciones tienen cardinalidad (cuántas entidades participan una relación) y condicionalidad (obligatoriedad o no). Por ejemplo:

En una zona puede haber cero, uno o muchos comercios, pero un comercio pertenece solamente a una zona (Nc – 1).

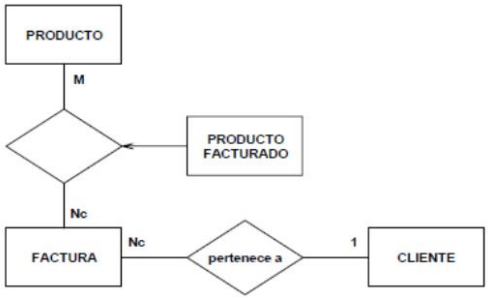
El modelo conceptual de datos, básicamente, lo que proponía era modelar entidades y relaciones. Esto es lo mismo que cuando estamos hablando de objetos y asociaciones entre objetos. Cada entidad es una abstracción, esto también está basado en teoría de conjuntos. La particularidad que plantea el diagrama de entidad-relaciones es que los objetos tienen que ser siempre instanciables, no puede ser que haya una sola instancia de un objeto, sino en ese caso no sería un objeto, una entidad en este modelo. Eso en el modelo de dominio no es algo que sea importante. Pero la verdad es que existen también los objetos débiles, que en definitiva son equivalentes a lo que representamos en el modelo de dominio con una asociación calificada. Cada entidad tiene atributos, algunos atributos son identificadores, y cada entidad debe tener siempre un identificador. Esa restricción en el modo dominio no existe.

Funciones y datos

Y después, sí empiezan los problemas, porque en el análisis estructura tradicional esta entidad-relación tenía que balancear con lo que se representa en los diagrama de flujo de datos como almacenamientos. Y acá es donde algunos autores no nos terminan de cerrar del todo mucho el tema. Hay 2 autores muy importante que vamos a nombrar más adelante, que son McMenamin & Palmer, y en realidad los datastore que aparecen en DFD[1] corresponden a los objetos del DER[2], no hay otra cosa para hacer ahí.

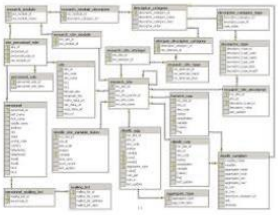


Nombre		Descripción
Cliente	=	@Nro Cliente + Razon Social Cliente + Direccion Cliente
Estadistica Ventas por Producto	=	Fecha Informe + {Codigo Producto + {Mes + Cantidad}}
Factura	=	@Nro Factura + Fecha Emision Factura
Factura Cliente	=	Nro Factura + Fecha Emision Factura + Razon Social Cliente + Direccion Cliente + {Codigo Producto + Cantidad Venta + Precio} + Total
Pedido Venta	=	Nro Cliente + {Codigo Producto + Cantidad}
Producto	=	@Codigo Producto + Descripcion Producto + Stock Actual Producto + Precio Producto
Producto Facturado	=	@Factura-ref-Nc + @Producto-ref-M + Cantidad Venta + Precio Venta



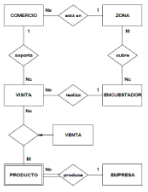
Modelado de datos y modelado de objetos

Tres comunidades relacionadas, pero con perspectivas diferentes

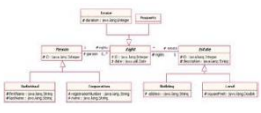


El tema se complica porque hay distintas comunidades:

- Por un lado está la gente que está más cerca de la **base de datos** que lo que le interesa más es trabajar en el espacio de la solución, con lo cual se empezó a utilizar en algunos casos una especie de entidad-relación pero orientado a la representación de tablas. Con lo cual dejó de usarse con el sentido que proponía Chen que era un modelo conceptual.



- Después está la comunidad que tiene que ver más con el **modelado conceptual de datos**, que le interesa trabajar en el modelado del espacio del problema, que cree que los datos son activos de organización.



- Por último, está la comunidad de **objetos**, que está más cerca de la solución y que para ellos, todo es un objeto.

[1] Data Flow Diagram
[2] Diagrama Entidad – Relación

Entonces hay 3 grandes tribus hablando de lo mismo o de cosas parecidas, y acá es muy difícil ponerse de acuerdo.

Conciliación de perspectivas

En realidad lo que dicen los autores es que UML se puede utilizar para modelar datos conceptuales (sería modelado de dominio), para modelar las base de datos (sería modelo lógico de la base de datos) , y para modelar objetos más cerca de lo que sería el diseño, en terminología del diseño de la solución. Hay que tener cuidado con las particularidades, pero se puede utilizar UML para las 3 tribus, pero hay que ponerse de acuerdo en algunas cosas básicas: Cuando estamos hablando de modelado de conceptos, estamos hablando de modelo de dominio, ahí no hay temas de implementación. Eso puede servir para que diseñemos una base de datos, puede servir para que diseñemos los objetos de diseño, los objetos de la solución pero, hay que entender que son vocabularios diferentes.

Conclusiones

Resumen

Los modelos entonces nos ayudan a entender el problema, a definir el sistema deseado, pero hay que tener cuidado con qué modelos tenemos que construir. No podemos avanzar ciegamente y construir todos los modelos que muchas veces nos proponen las diversas metodologías que andan dando vueltas, porque no siempre va a ser necesario. Hay infinitas variables que van a influenciar: la complejidad del problema, la complejidad de la solución, el tipo de proyecto, la cantidad de gente involucrada, la complejidad técnica, la complejidad de gestión, etc.

En particular, el modelado de dominio ya sea que lo hagamos del otro lado de una servilleta, en un pizarrón, o que lo hagamos formalmente en una herramienta de modelado, es fundamental para entender el vocabulario del dominio, para entender el contexto en el que va a operar el futuro sistema y para derivar una buena solución del diseño.