9. Modelo de dominio

Intro

Veremos diferentes modelos (De dominio y de diseño), nos interesa el de diseño.
Vemos diferentes mecanismo para expresar el modelo de dominio, de las cuales solemos aplicar patrones de análisis:

1. usar *analisis sintáctico* para encontrar los actores, lugares y eventos que nos interesan.
2. usar *CRUD* para poder verificar si a cada entidad se le aplican las 4 operaciones CRUD, si no es el caso probablemente nos falte una funcionalidad
   Luego de ello buscamos los patrones de colaboración, en particular en las transacciones que representan los eventos ubicados en el analisis sintpactico, pero viendo cómo colaboran con los demás actores y en qué patrón de colaboración encaja.
   Teniendo el gráfico con los patrones de colaboración integrados, lo unico que falta son las reglas de negocio, sin estas reglas el modelo de dominio se encuentra incompleto. Para esto colocamos en texto a regla de negocio que se debe cumplir (lo más cerca posible de la entidad que dispone de más información relevante de dicha regla de negocio)

COMPLEJIDAD DEL DESAROLLO DE SOFTWARE

Antes de plantear solcuiones a un problema, hay que entenderlo.
Podemos categoriza la complejidad del problema como:

> - complejidad = complejidad escencial + complejidad accidental

- complejidad escencial = complejidad del problema + complejidad de la solcuión

Podemos atacar la complejidad a través de *mecanismos* (POO):

- Descomposición (descurbrir)
- Abstracción (pensar desde qué perspectiva vemos las entendidades los comportamientos de interés)
- Establecer jerarquías (inventar nuevas soluciones: ejm controladores)

Modelos: de dominio y de diseño

Modelos

Usamos **modelos** como una representación *simplificada* de la realidad. Estos permiten:

- visualizar
- entender
- especificar

Sirven de guía y documentación.

MODELOS DE DOMINIO

- representación conceptual del problema a resolver, se quiere entender en detalle las reglas de negocio. Se definen las entidades de dominio, las reglas y conceptos relevantes en estos. Por ejemplo: cliente (no menor de 18), reserva (con fecha posterior a la actual), habitación (existente en el hotel)
- Se construye a *partir de los casos de uso*.
- Se divide en dos partes:
  - *comportamiento*: la lógica del sistema, cómo interactúan diferentes instancias en ciertos casos -> diagramas de secuencia o de colaboración
  - *estructura*: organización estática de las clases y sus relaciones (no hablamos de objetos, solo clases). -> Diagramas de clases.
- los mecanismos empleados son:
  - Patrones de análisis (análisis sintáctico)

- Patrones de colaboración

MODELO DE DISEÑO

- el modelo de diseño agrega los detalles técnicos al modelo de dominio para convertirlo en código. Aquí aparecen entidades como *clases controladoras de otras*, dependencias, *patrones de diseño* (como MVC, Strategy, etc).
- el modelo de diseño se orienta en plantear soluciones al modelo planteado teniendo en cuenta las restricciones del modelo de dominio y los *requerimientos no funcionales*

Modelo de dominio: patrones de análisis -> patrones de colaboración

Mecanismos usados: Patrones de análisis

ANALISIS SINTÁCTICO

- mecanismo para obtener el **modelo de dominio** a partir de textos.
- consiste en extraer actores, operaciones y relaciones a partir de los requerimientos escritos como historias de usuario o descripciones en lenguaje natural (minutas)
- Se analizan sustantivos y verbos significativos y las categorizamos como:
  - actores (humanos)
  - objetos físicos
  - lugares
  - eventos
  - procesos
- por ejemplo:
  - user story: "Como cliente, quiero reservar una habitación para una fecha específica."
  - clases del modelo de dominio:
    - cliente (actor)

- habitación (lugar)
- reserva (evento)

CRUD:

- Define las operaciones básicas que se pueden hacer sobre una entidad del dominio: Create, Read, Update, Delete.
- Nos ayuda a definir las funcionalidades **mínimas** sobre una entidad.

Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

  - Toda entidad debe ser creada y leida en alguna funcionalidad
  - Suele modificarse o eliminars
  - si falta una operacion: probablemente fata una funcionalidad

Mecanismos usados: Patrones de colaboración

- Describen cómo interactúan objetos entre sí para lograr un comportamiento. Por ejemplo:
  - un cliente realiza una reserva en una habitación en cierta fecha
- Útil para saber cómo las entidades del modelo de dominio interactuan entre sí.
- Hay 3 grupos de patrones de colaboración, entre algunos de ellos podemos armar transacciones simples, transacciones compuestas y transacciones cronológica cuando conocemos la cronología.

PATRONES DE COLABORACIÓN AGRUPADOS:

1. *Genérico-específico:*
   - actor - rol
     - un actor puede conocer múltiples roles pero solo juega uno.

    - toda acción que tome un actor será en el contexto de un rol.
    - un rol puede ser a su vez el actor en referecia a oro rol más específico.
- item - item específico:
    - el ítem describe la información útil en todas las variantes.
    - la variante describe lo que la especializa
    - la variante es la que paticipa en eventos

> ⚠ **parte de una transacción compuesta**
>
> - transacción compuesta - line item

2. *entero-parte:*

- gran lugar -lugar:
    - modela la relación entre el lugar donde ocurre "el evento" y el lugar al que este pertenece (a nivel jerarquico)
- ensamble-parte
    - en ensamble se compone de 1 o más componentes.
- contenedor-contenido
    - un recipiente puede guardar muchas cosas (diferentes contenidos) o estar vacio
- grupo-miembro
    - modelan clasificaciones de cosas, personas, lugares.
    - pueden haber clasificaciones sin miembros aun agregados
    - a diferencia de contenedor-contenido, los miembros pueden pertenecer a más de una grupo

3. *específico-transacción:*

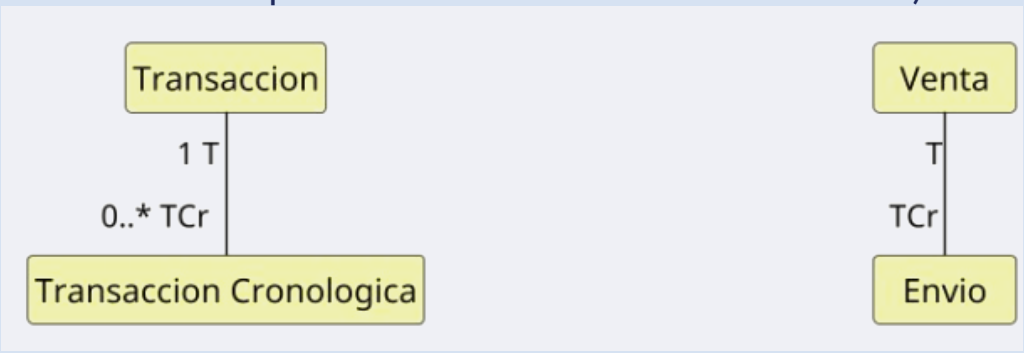
> ⚠ **parte de una transacción compuesta**
>
> - item específico - *line item*

> **parte de una transacción simple**
>
> - Rol - Transacción

    - entidad interactuando con algo en algún lugar
- item específico-transacción
- lugar-transacción

transacción-transacción cronológica

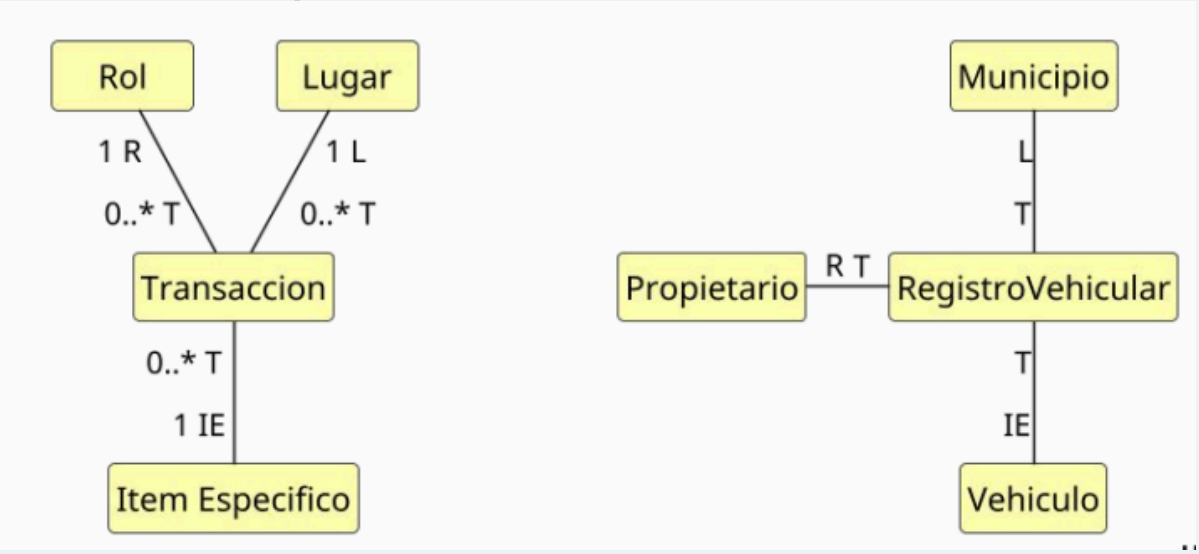
- nos permite modelar transacciones que siguen a interacciones anteriores (relaciona dos transacciones y la que tiene el extremo T es la que debe suceder antes del extremo TCr)



TRANSACCIONES:

Transaccciones simples:

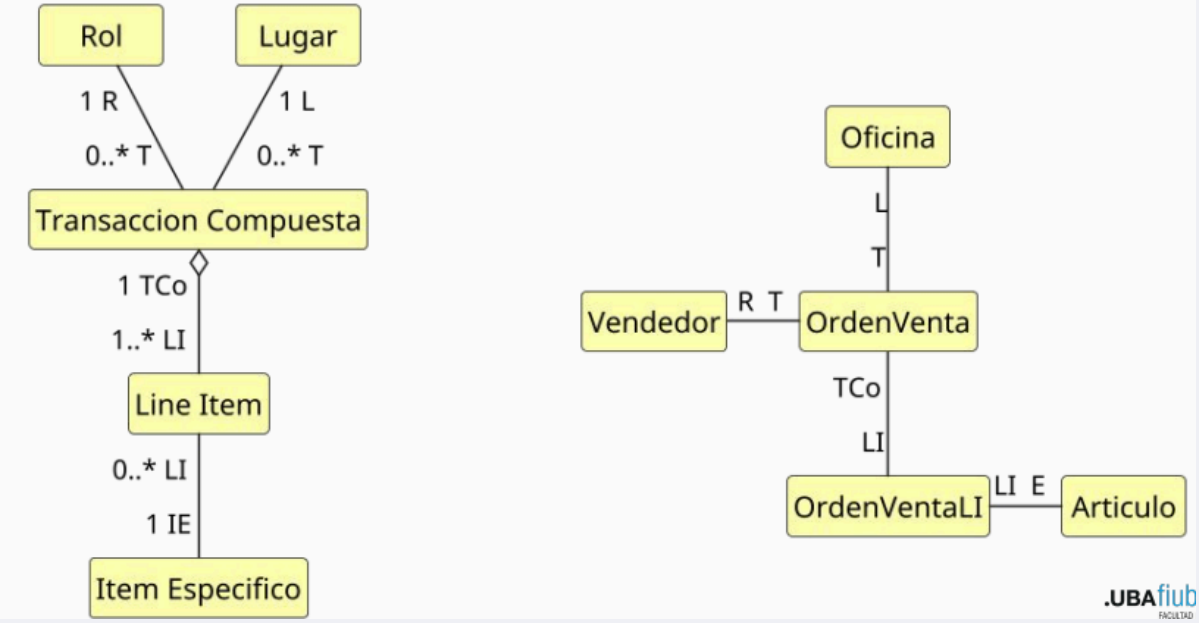
- Una transacción conoce quién la realiza (rol), dónde (lugar) y sobre qué se realiza (**ítem específico**)



Transacciones complejas:

- Describe la interacción de una persona en un lugar con 1 o varias líneas de items, donde *cada línea de ítem puede relacionarse con un ítem específico*.

- Por ejemplo: una vendedor en una oficina que realiza la transacción "orden de venta" que involucra una orden de venta particular para un lineitem que se asocia con un articulo.



- el Line item solo existe si es que se relaciona cuando menos con una transacción compuesta, en esta transacción se realiza con un item específico.
- el line item es la forma de agregar mutltiplicidad

+ Reglas de negocio

- Son *restricciones que rigen en el dominio de negocio*. aseguran el sistema funcione correctamente y refleje la realidad del negocio.
- En el modelo se traducen a **reglas de colaboración**, el incorporar una regla de negocio implica aplicar restricciones que se deben de cumplir cada que se quiera modificar la colaboración entre los distintos objetos del modelo.

Tipos de reglas

- Tipo: un medicamente solo puede encargarse en un container refrigerado (el tipo del container debe ser refrigerado)
- Multiplicidad: un pallet refrigerado puede contener hasta 10 caja (un límite numérico)

- Propiedad: un pago debe registrar un número válido de tarjeta de crédito (cualidad del número provisto, no es un tipo)
- Estado: una orden cancelada no se debe de entregar (el estado de la orden define si se entrega o no)
- Conflicto: un producto con alcohol no puede agregarse al carrito de un comprador si es menor de edad.