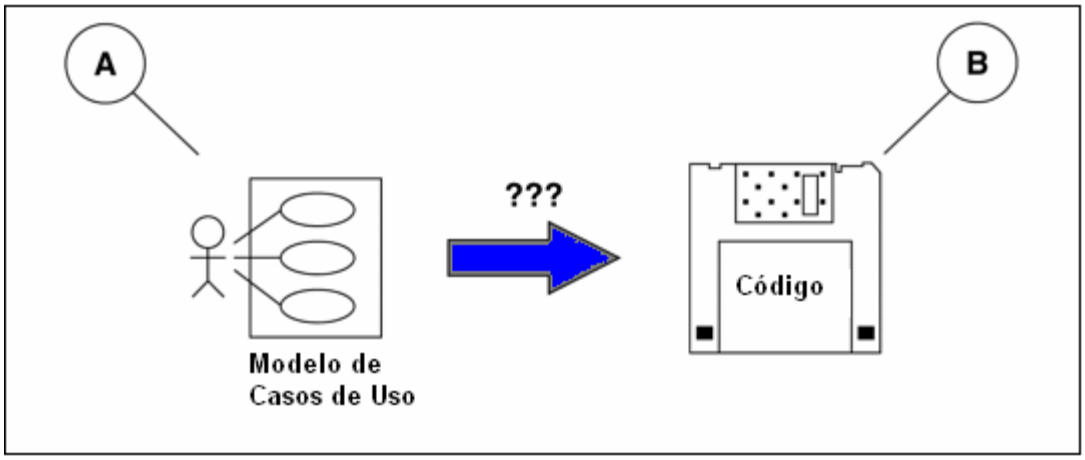
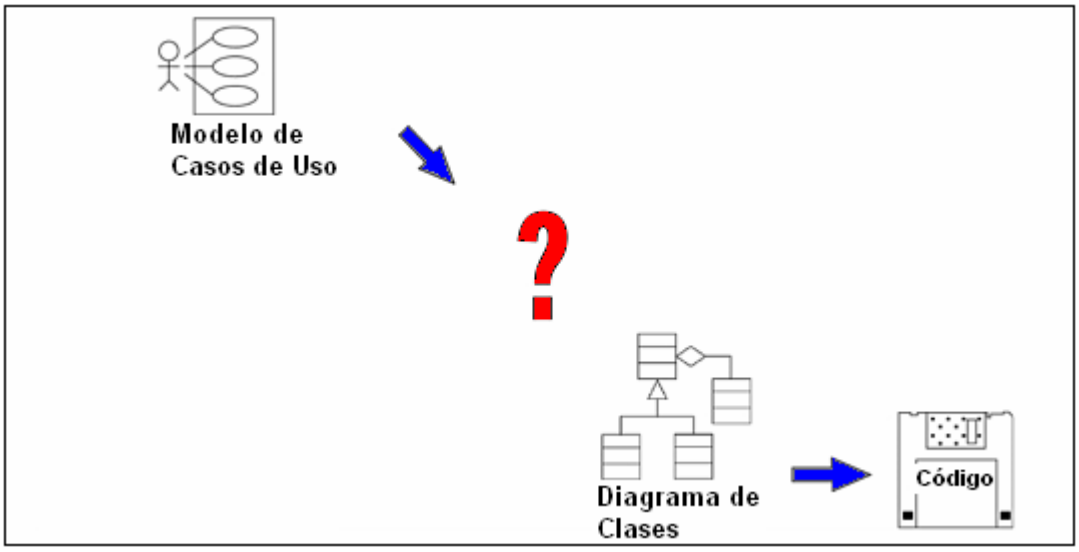
it-Mentor

Patrones de Análisis

Proceso de desarrollo – Relación entre modelos

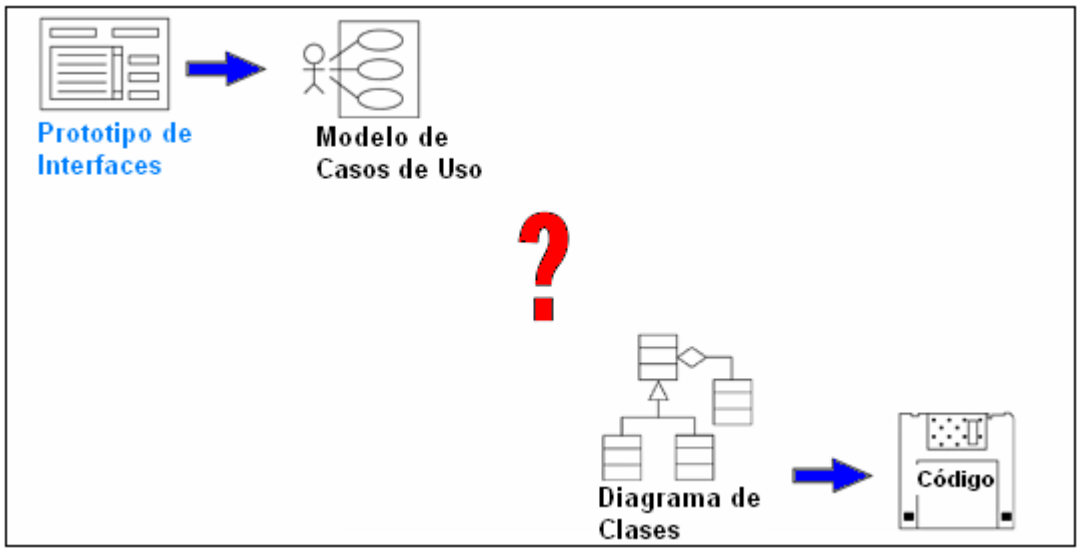


Objetivo: convertir en un sistema los requerimientos

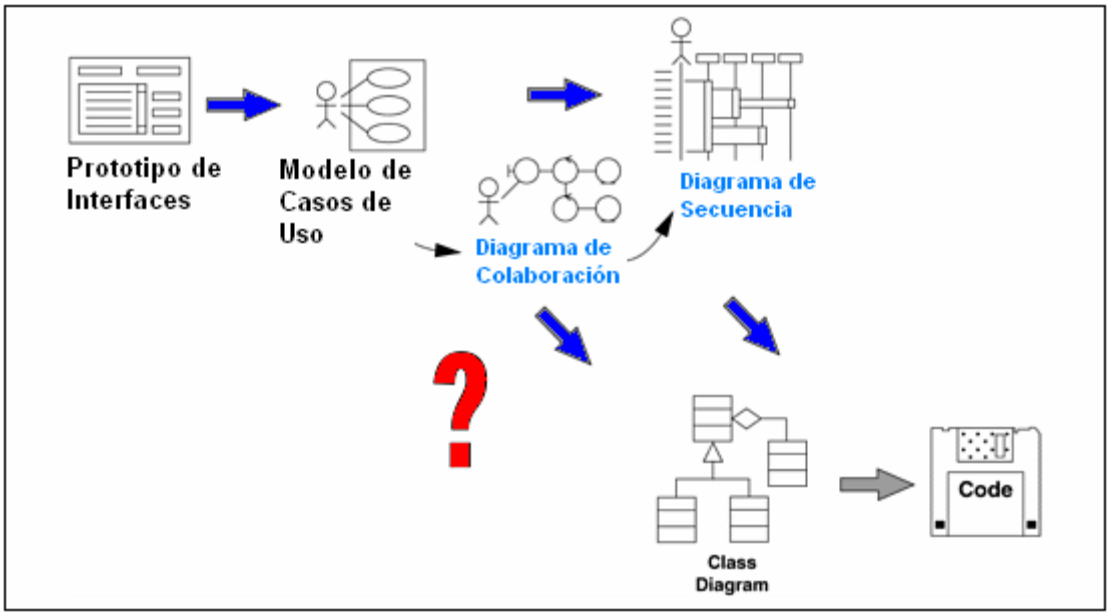


Sabemos que el código surgió de un diseño previo. Pero, **¿Cómo surgió el diseño?**

Seguimos la secuencia de pasos que nos lleva a este Diseño.

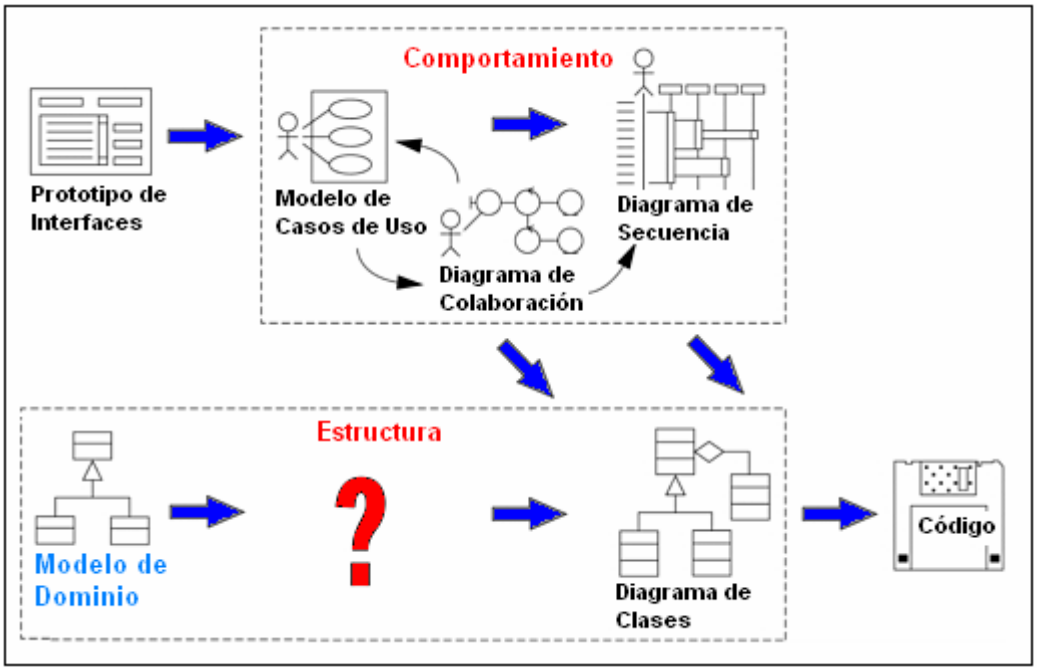


Primer 1: relevar requerimientos a partir de las necesidades de los usuarios en relación con su trabajo utilizando el sistema en desarrollo.

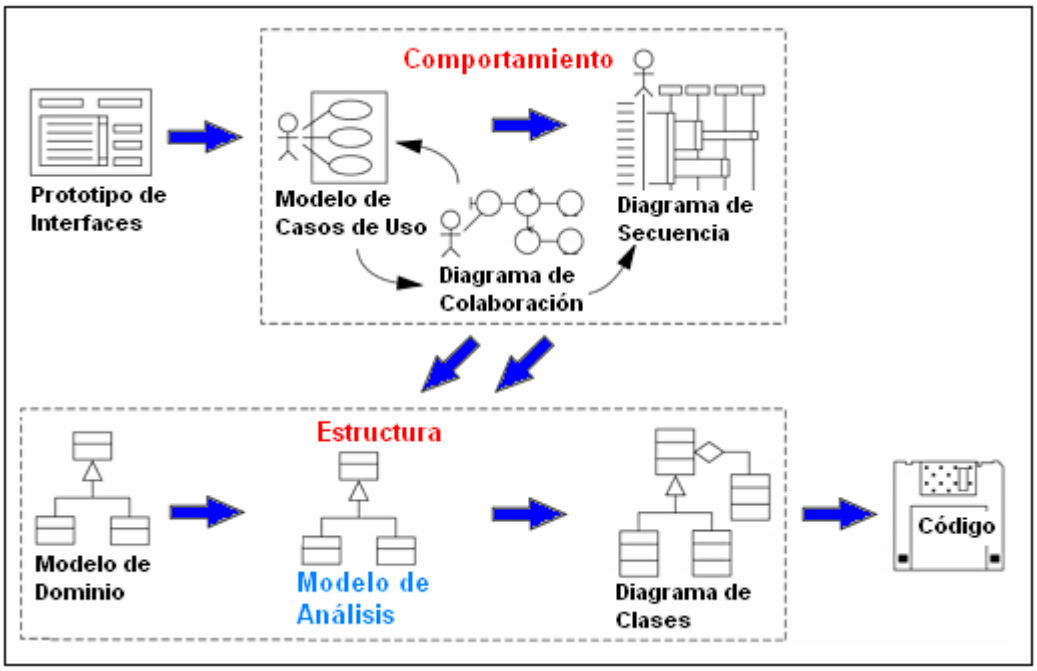


Paso 2: especificar los casos de uso para entender su comportamiento dinámico (diagramas de colaboración y / o secuencia).

Tenemos Comportamiento, hace falta una estructura.



Paso 3: pensar una estructura simple a partir de conceptos extraídos del dominio del problema (Modelo de Dominio).



Paso 4: refinar el Modelo de Dominio a partir de la información de los Casos de Uso y Patrones de Análisis en un Modelo de Análisis.

MODELO DE ANÁLISIS

- Objetivo: entender en detalle el negocio y sus reglas
- Mecanismo utilizado: Patrones de Análisis

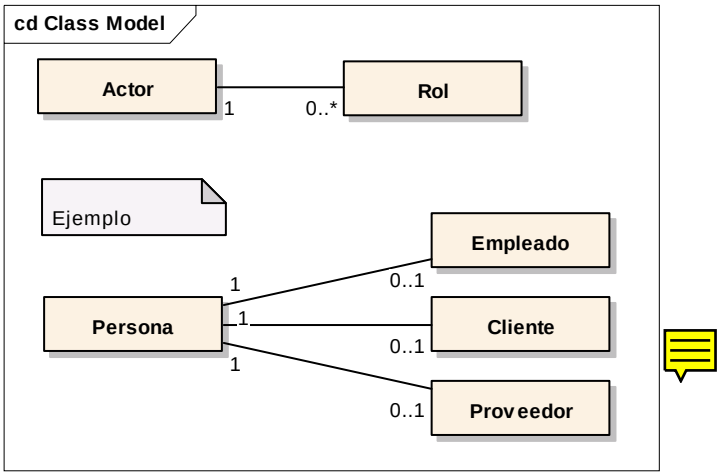
MODELO DE DISEÑO

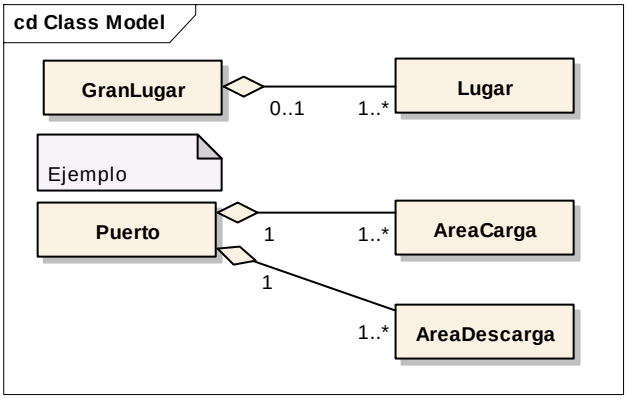
- Objetivo: implementar una solución al problema planteado en el análisis más las restricciones impuestas por los requerimientos no funcionales.
- Mecanismo utilizado: Patrones de Diseño

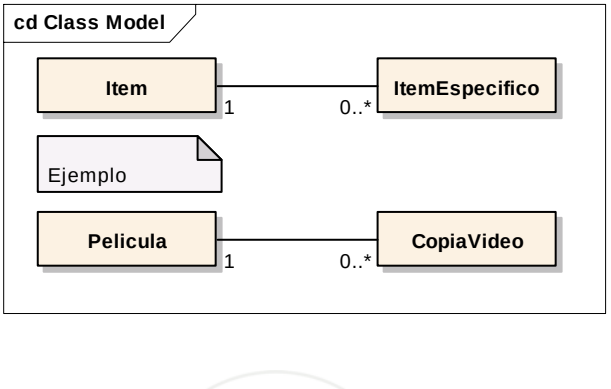
Modelo de Análisis - Guía de selección de objetos

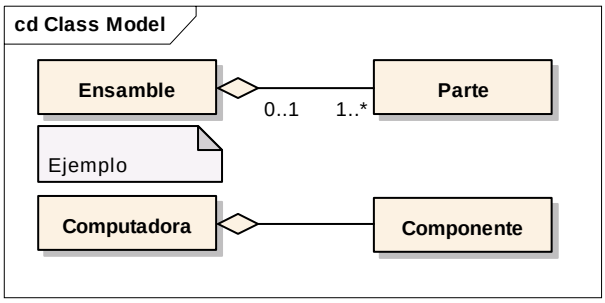
Conceptos a buscar	Instancias
Gente	Actor Rol
Lugares	Lugar Gran Lugar
Cosas	Ítem Ítem Específico Ensamble Parte Contenedor Contenido Grupo Miembro
Eventos	Transacciones Transacciones Compuestas Transacciones Cronológicas Line Ítem

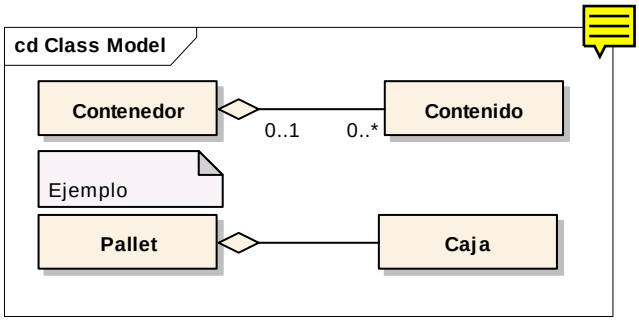
PATRONES DE COLABORACIÓN ENTRE OBJETOS

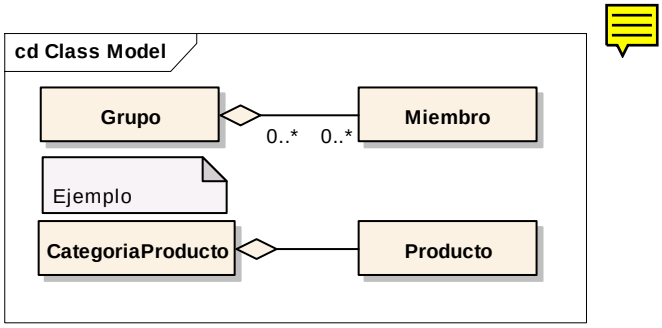

cd Class Model
Actor
1
0..*
Rol
Ejemplo
Empleado
1
0..1
Persona
1
Cliente
1
0..1
0..1
Proveedor

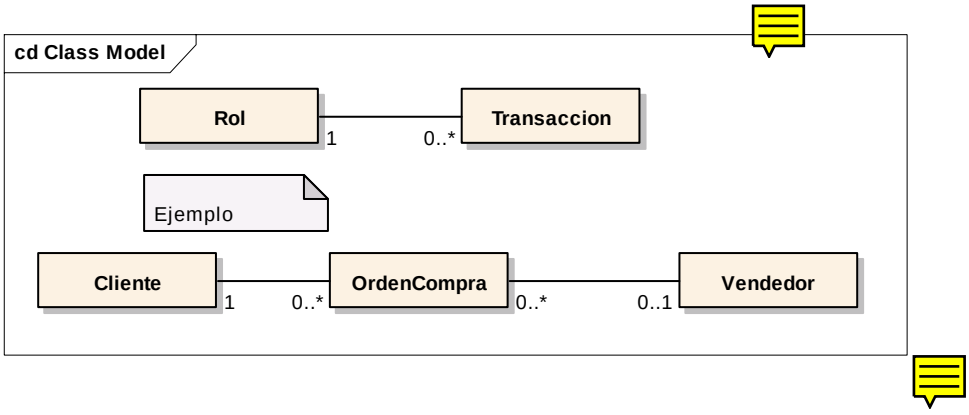

cd Class Model
GranLugar
0..1
1..*
Lugar
Ejemplo
Puerto
1
1..*
AreaCarga
1
1..*
AreaDescarga

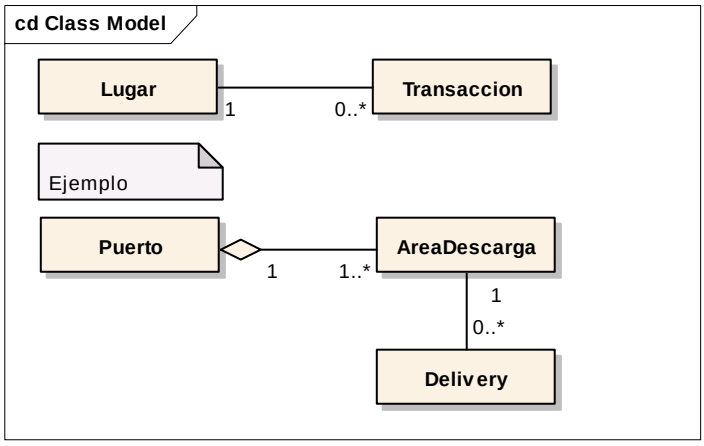

cd Class Model
Item
1
0..*
ItemEspecifico
Ejemplo
Pelicula
1
0..*
CopiaVideo

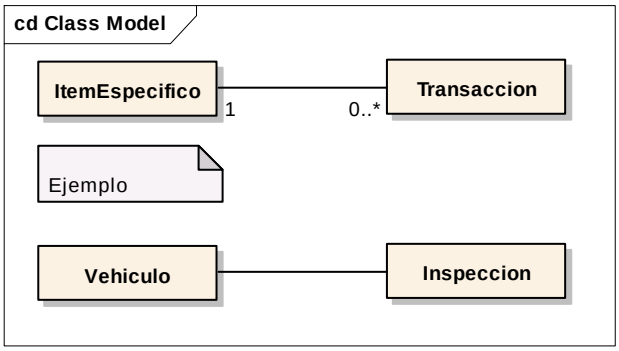

cd Class Model
Ensamble
0..1
1..*
Parte
Ejemplo
Computadora
Componente

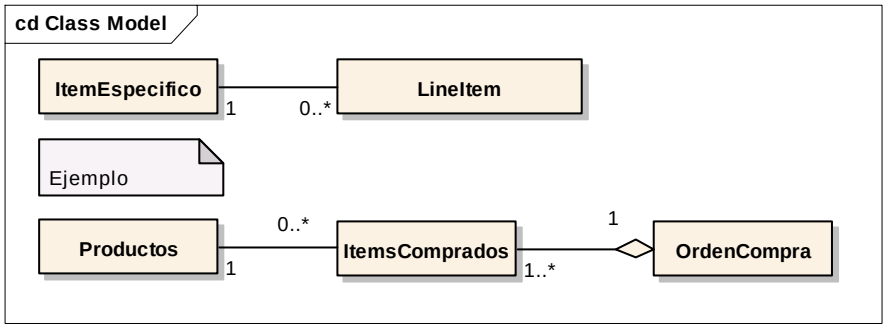

cd Class Model
Contenedor
0..1
0..*
Contenido
Ejemplo
Pallet
Caja

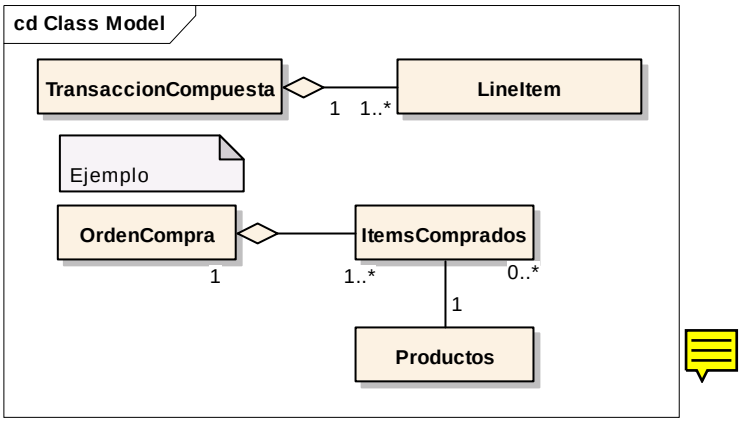

cd Class Model
Grupo
0..*
0..*
Miembro
Ejemplo
CategoriaProducto
Producto

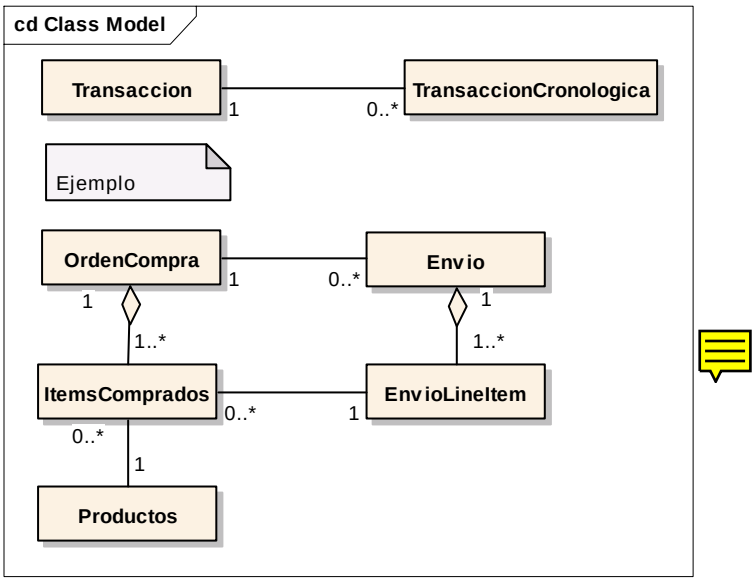

cd Class Model
Rol
1
0..*
Transaccion
Ejemplo
Cliente
1
0..*
OrdenCompra
0..*
0..1
Vendedor


cd Class Model
Lugar
1
0..*
Transaccion
Ejemplo
Puerto
1
1..*
AreaDescarga
1
0..*
Delivery


cd Class Model
ItemEspecifico
1
0..*
Transaccion
Ejemplo
Vehiculo
Inspeccion


cd Class Model
ItemEspecifico
1
0..*
LineItem
Ejemplo
Productos
0..*
1
ItemsComprados
1
1..*
OrdenCompra


cd Class Model
TransaccionCompuesta
1
1..*
LineItem
Ejemplo
OrdenCompra
1
ItemsComprados
1..*
0..*
1
Productos


cd Class Model
Transaccion
1
0..*
TransaccionCronologica
Ejemplo
OrdenCompra
1
1
0..*
Envio
1
1..*
1..*
ItemsComprados
0..*
1
EnvioLineItem
0..*
1
Productos

REGLAS DE NEGOCIO

Son las restricciones que gobiernan las acciones dentro de un dominio de negocio.

- En el modelo se traducen en reglas de colaboración.
- La forma de incorporarlas al modelo consiste en restricciones a ser probadas en la colaboración entre los distintos objetos del modelo.
- En el modelo se traduce por ejemplo en si dos objetos pueden crear una nueva relación o remover una existente.
- ¿Donde ubicarlas?
- Si no se ubican en el modelo, el mismo es incompleto. La dinámica de los objetos será externa al mismo.
- ¿Cómo expresarlas? ¿Qué colaborador (objeto) prueba cada regla en función de la información que conoce o puede consultar?

Tipos de reglas

- **Tipo**: un medicamento puede ser cargado solo en un container refrigerado
- **Multiplicidad**: un pallet refrigerado puede contener hasta 10 cajas
- **Propiedad** (validación, comparación): un pago debe registrar un número válido de tarjeta de crédito, la temperatura de un container refrigerado debe ser menor a 0 grados centígrados
- **Estado**: una orden no debe ser entregada si antes fue cancelada
- **Conflicto**: un vuelo no puede ser programado en una puerta en un mismo horario que otro vuelo, un producto no puede ser sumado a una orden de compra de un menor de edad si la venta está prohibida a menores.

Reglas de colaboración: chequeo de reglas de negocio entre objetos participantes de las relaciones. El cambio de estado, el establecimiento de una relación o la ruptura de la misma requiere el chequeo de una regla de negocio. Este chequeo lo vemos como una colaboración entre objetos.

Asignación

Asignación de Reglas:

1. cuando se modela gente, lugares y cosas, SIEMPRE asignar las reglas a los objetos más específicos, locales o detallados.
2. cuando la gente, lugares y cosas interactúan con un evento, son los propietarios de sus reglas.
3. cuando existen transacciones en secuencia, la transacción precedente es la que chequea las reglas.

Regla	Patrón de colaboración											
	A-R	GL-L	I-IE	E-P	C-C	G-M	T-R	T-L	T-IE	TC-LI	LI-IE	TC-T
Tipo	R	L	IE	P	C	M	R	L	IE	LI		T
Multiplicidad	R	GL, L	I, IE	E, P	C, C	G, M	T, R	T, L	T, IE	TC, LI	LI, IE	TC, T
Propiedad	R	GL, L	IE	E, P	C, C	G, M	R	L	IE			T
Estado	R	GL, L	I, IE	E, P	C, C	G, M	R	L	IE			T
Conflicto	R	L	IE	P	C	G, M	R	L	IE			T

4. En el patrón A-R las reglas dependen del contexto, por esta razón están asignadas al rol, ya que este es quien conoce el contexto.
5. Los patrones de agregación (GL-L, E-P, C-C, G-M) tienen un comportamiento similar en cuanto a las reglas que deben probar. Solo en el G-M el G debe probar la regla que define el conflicto con otros grupos.
6. Los patrones en los que uno de los objetos es un evento o transacción (T-R, T-L, T-IE) la responsabilidad de la prueba de las reglas es asignada a los objetos que participan de la transacción. La transacción debe cumplir con que solo conoce al colaborador y no puede romper con él. Los objetos que participan pueden ser interrogados acerca de si aceptan participar de la transacción, en ese momento prueban las reglas. (ver servicios que conducen). En el caso de las transacciones cronológicas se da el mismo patrón, donde la precedente adopta el rol de la prueba de reglas.
7. En el caso de transacciones secuenciales o cronológicas (TC-T), la precedente prueba las reglas. Solo la multiplicidad es chequeada por la segunda transacción.

Patrones de asignación de Reglas de Negocio

Reglas asociadas a Propiedades de las clases

⇒ **Tipos de Reglas**

Tipo, Propiedades, Fechas, Horas, Descripciones, Estados

⇒ **Patrón**

- **setXXX(valor)** solo estos métodos aparecen en las interfaces
- **testXXX(valor)**
- **doSetXXX(valor)**

Ej. SetTitle(String) en Documento

Ej. makeChair() en TeamMember (no se usa setChair porque se reservan los sets para métodos con valores).

⇒ **Validación**

La validación la realiza la clase propietaria de la propiedad

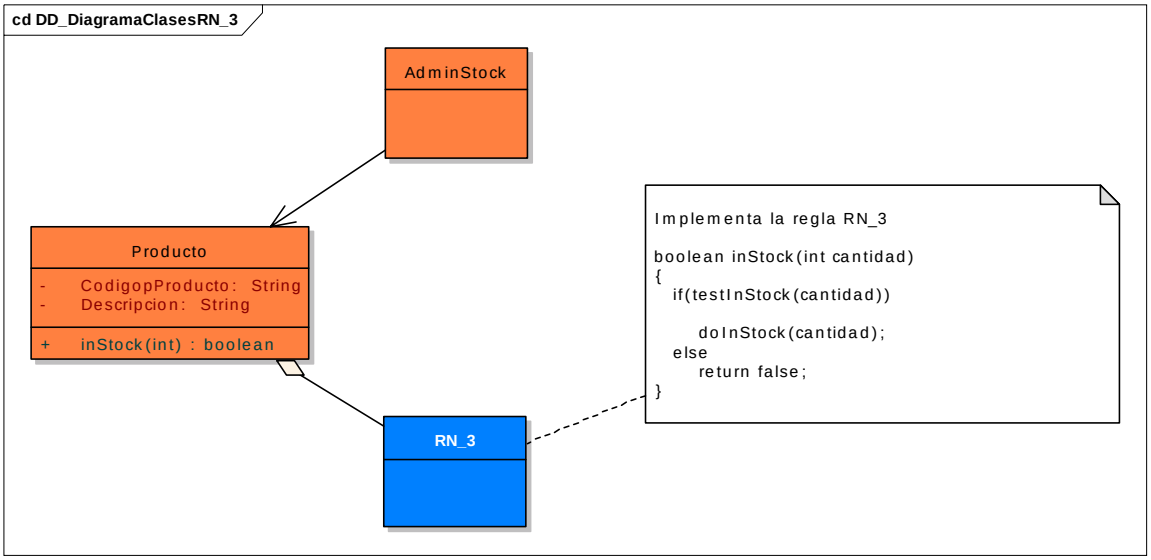
⇒ **Variaciones**

Validaciones cruzadas

Una propiedad de una clase depende de otra clase (Ej. TeamMember – Team)

Ej. testSetRoleChair() en TeamMember

Ej. Implementación



Reglas asociadas a Colaboraciones

⇒ **Tipos de Reglas**

Reglas que son validadas con la participación de más de una clase, los que forman el patrón de análisis.

⇒ **Patrón**

- **addXXX(referencia)**
- **testAddXXX(referencia)**
- **doAddXXX(referencia)**

- **removeXXX(referencia)**
- **testRemoveXXX(referencia)**
- **doRemoveXXX(referencia)**

⇒ **Validación**

Asignar validación y director a las colaboraciones

Las validaciones se delegan:

Genérico -> Específico
Todo -> Parte
Específico -> Transacción

Las validaciones las conducen:
Genérico – Específico
Todo – Parte
Específico – Transacción

- ✓ Los métodos addXX y removeXX solo aparecen en las interfaces de las clases que conducen las validaciones
- ✓ Los métodos doAddXX, doRemoveXXX y los testAddXX, testRemoveXXX aparecen en todas

Ej. TeamMember – Nominacion – Documento

Reglas asociadas a Conflictos

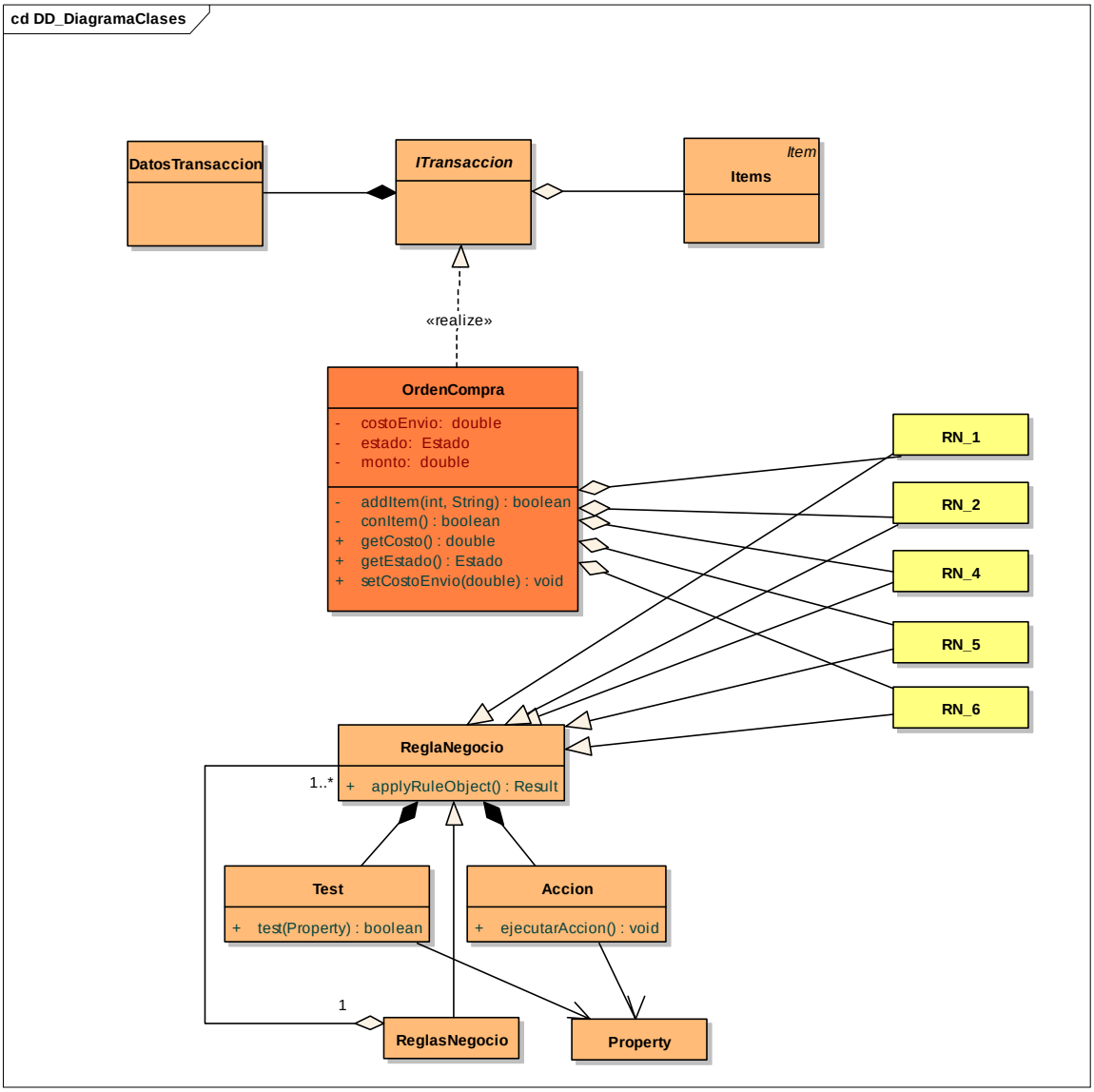
⇒ **Tipos de Reglas**
Conflicto

⇒ **Patrón**

⇒ **Validación**
El director las valida desde sus propias validaciones

Ej. Nominacion TestAddDocumento() –> Documento
testAddNominationConflic()

Ej. Implementación de Reglas compuestas



TIPOS DE SERVICIOS

1. conducción del negocio
2. interrogación (información del estado actual)
3. análisis de transacciones (información histórica)

Asignación

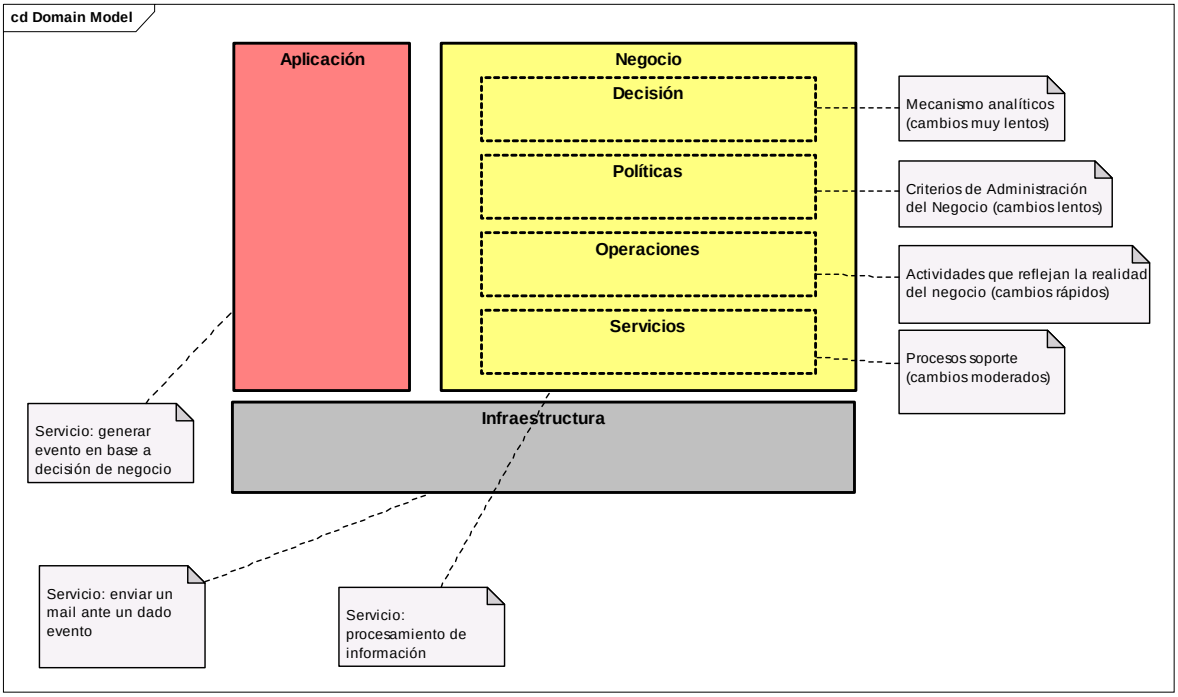
Asignación de servicios:

1. Los objetos más específicos conducen la realización del negocio a partir de sus servicios.
2. cuando para la realización de una acción es necesario la participación de más de un objeto, los eventos conducen a través de sus servicios.

Esquema de una clase		
Tipo de Servicios	**Comentarios**	**Tipo de métodos comprendidos**
Servicios de Conducción del Negocio	Acciones: creación de objetos, asignación de valores. Cambian estados, establecen relaciones y las rompen.	**void makeChair()**, *ver TeamMember*
		void grantNominatePrivilege(), *ver TeamMember*
		void nominate(ITeamMember), *ver Documento*
Servicios de Determinación de Valores	Respuestas a preguntas: información actual, no cambia estados, devuelve valores o los calcula en colaboración.	**boolean hasNominatePrivilege()**, *ver TeamMemberProfile*
		boolean hasValidEmail(), *ver PersonProfile*
		boolean isPublished(), *ver Documento*
Servicios de Análisis de Transacciones	Respuestas a preguntas: información histórica	**INomination getApprovedNomination()**, *ver Documento*
		INomination getLatestNomination(), *ver Documento*
Acceso y asignación a Propiedades	Propiedades: acceso a propiedades para obtener y asignar	**String getTitle(),** *ver Documento*
		void setTitle(String), *ver Documento*
Suma y remoción de Colaboradores	*Colaboradores:* establecen relaciones y las rompen.	**void addPerson(Iperson),** *ver TeamMember*
		void removePerson(Iperson), *ver TeamMember*
Acceso a Colaboradores	*Colaboradores*: acceso a colaboradores	**IPerson getPerson(),** *ver TeamMemberProfile*
Reglas de Colaboración	*Reglas de Negocio:* validación de las reglas asignadas	**void testAddPerson(IPerson aPerson),** *ver TeamMember*
		Void testAddNominationConflict(INomination aNomination, ITeamMember aTeamMember), *ver Documento*

Transición Análisis - Diseño

- Definir entidades
- Definir value objects
- Delimitar agregaciones
- Encapsular en paquetes / módulos
- Aísle el Negocio de la aplicación
- Particionar el negocio en capas
- Delimitar contextos
- Segregar y abstraer la esencia (core)
- Delimitar sub. dominios
- Definir servicios



Referencias

BIBLIOGRAFÍA

Libros

- *Streamlined Object Modeling – Patterns, rules and implementations,* Jill Nicola et. al., PHPTR, 2002.
- *Analysis Patterns*, Martin Fowler, 1997.
- *Domain Driven Design,* Eric Evans, Addison-Wesley, 2004.
- *Object-Oriented Analysis and Design with Applications (2nd Edition)*, Grady Booch, 1994.