

# Desarrollo de Software Dirigido por Modelos (DSM)

## Tema 1. Modelos y Metamodelos

---

ETS Ingeniería Informática - UPV  
Grado en Ingeniería Informática  
2016/2017



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

*DSIC*  
DEPARTAMENTO DE SISTEMAS  
INFORMÁTICOS Y COMPUTACIÓN

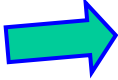
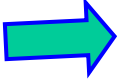






# Contenido

- I. Modelos y Metamodelos
- II. MOF y metamodelado
- III. Transformaciones

# De la Tecnología de Objetos a la Tecnología de Modelos

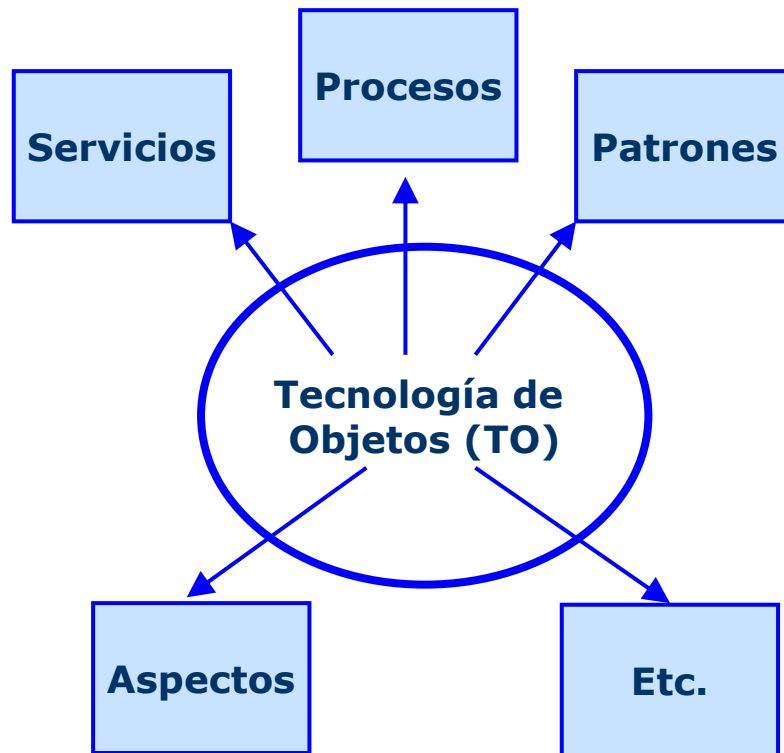
- El nuevo papel de los **modelos** (y metamodelos) en el ciclo de vida del software:
  - Del desarrollo **basado** en modelos al desarrollo **dirigido** por modelos
  - De la **composición** de objetos/componentes a la **transformación** de modelos
  - De la **Programación Orientada a Objetos** a la « **Programación** » con modelos
    - Programas = modelos + transformaciones  
(como análogamente tenemos el *código fuente* + *compilador*)
  - De lo *implícito* a lo *explícito*

# Importancia dinámica de los paradigmas en la IS

- Objetos 
- Componentes 
- Procesos   

- Servicios   

- Software as a Service (SaaS) (cloud computing)   


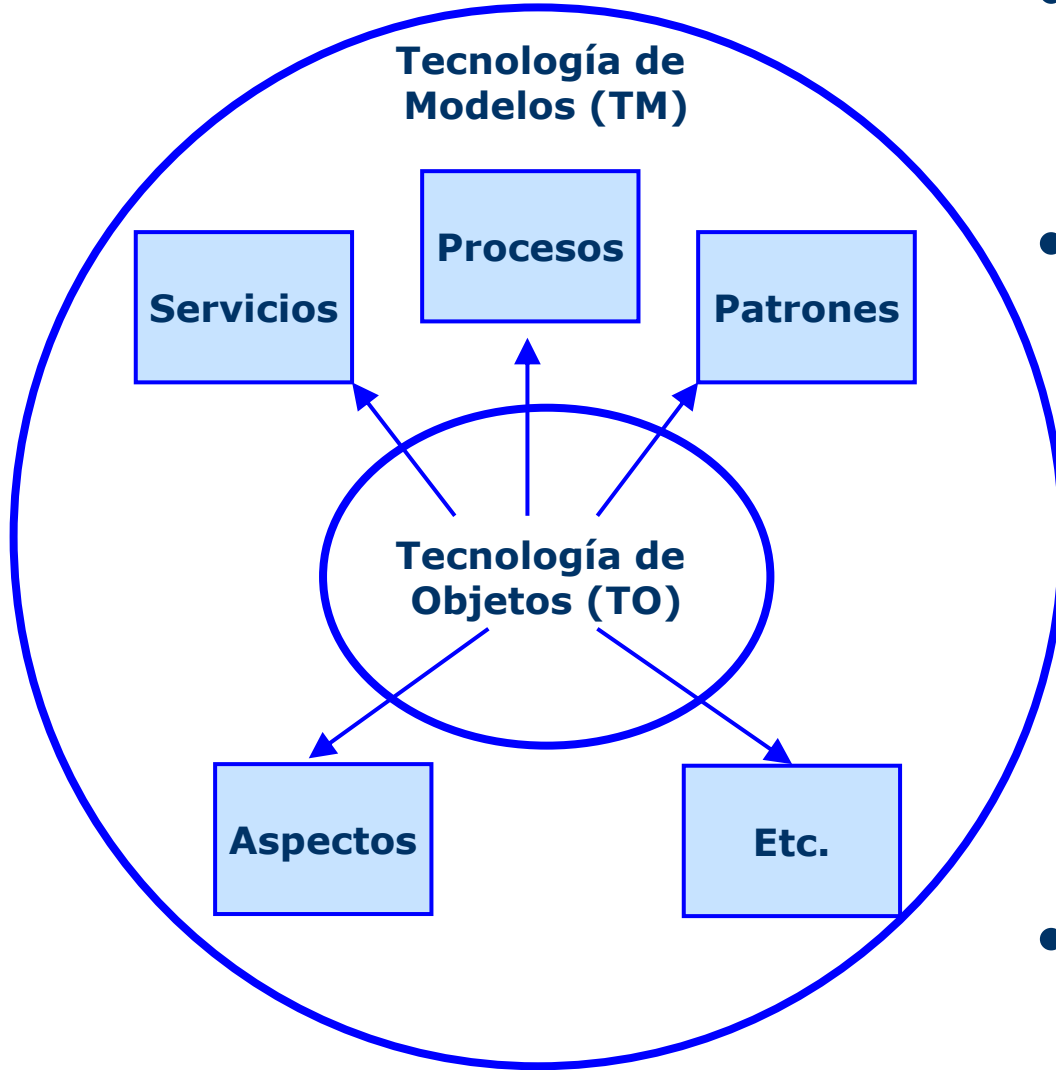
**La Tecnología de Modelos es capaz de incorporar todos estos paradigmas (y otros)**

# Con la Tecnología de Objetos



- La Tecnología de Objetos (TO) ha sido la respuesta para muchos problemas complejos desde su uso industrial en los 80s y 90s.
- Aunque generalmente, no era la propia Tecnología de Objetos quien proporcionaba la solución a estos problemas (**facilitador**).
- Con la POO, existen **problemas intrínsecos** que hacen que muchas soluciones sean *ad-hoc, informales o barrocas*.  
Son difíciles de *generalizar* o *combinar* o tiene problemas de *escalabilidad*.

# Con la Tecnología de Modelos



- La tecnología de Modelos (TM) parece estar en posición de abordar algunos de esos **problemas intrínsecos**.
- En otros:
  - Alto nivel de abstracción (problemas +complejos +grandes)
  - Separación de aspectos (tecnología, conocimiento)
  - Integración de diferentes puntos de vista: estructura, comportamiento, ...
  - Tratamiento homogéneo de conocimiento, arquitecturas, procesos,...
- TM parece capaz de “agregar” la tecnología de objetos y otras tecnologías ofreciendo una evolución realística a organizaciones y a problemas más ambiciosos

# ¿Es necesario este cambio de paradigma?

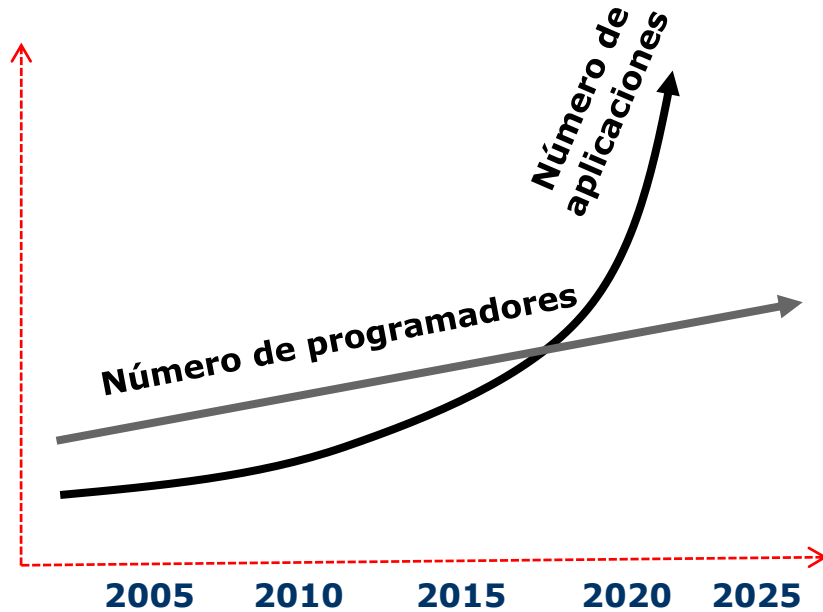
La tecnología crece a un ritmo exponencial...



- IoT, BigData, Agentes Autónomos (robots, drones, vehículos sin conductor), etc.
- Para 2020 → 1.7 MBytes generados persona/segundo → 44 trillones Gbytes
- Economía, ej. Ingresos por apps (app stores): 50.9\$ billones en 2016 hasta 101.1\$ billones en 2020 (venturebeat.com)

# ¿Es necesario este cambio de paradigma?

La tecnología crece a un ritmo exponencial...



La ecuación imposible

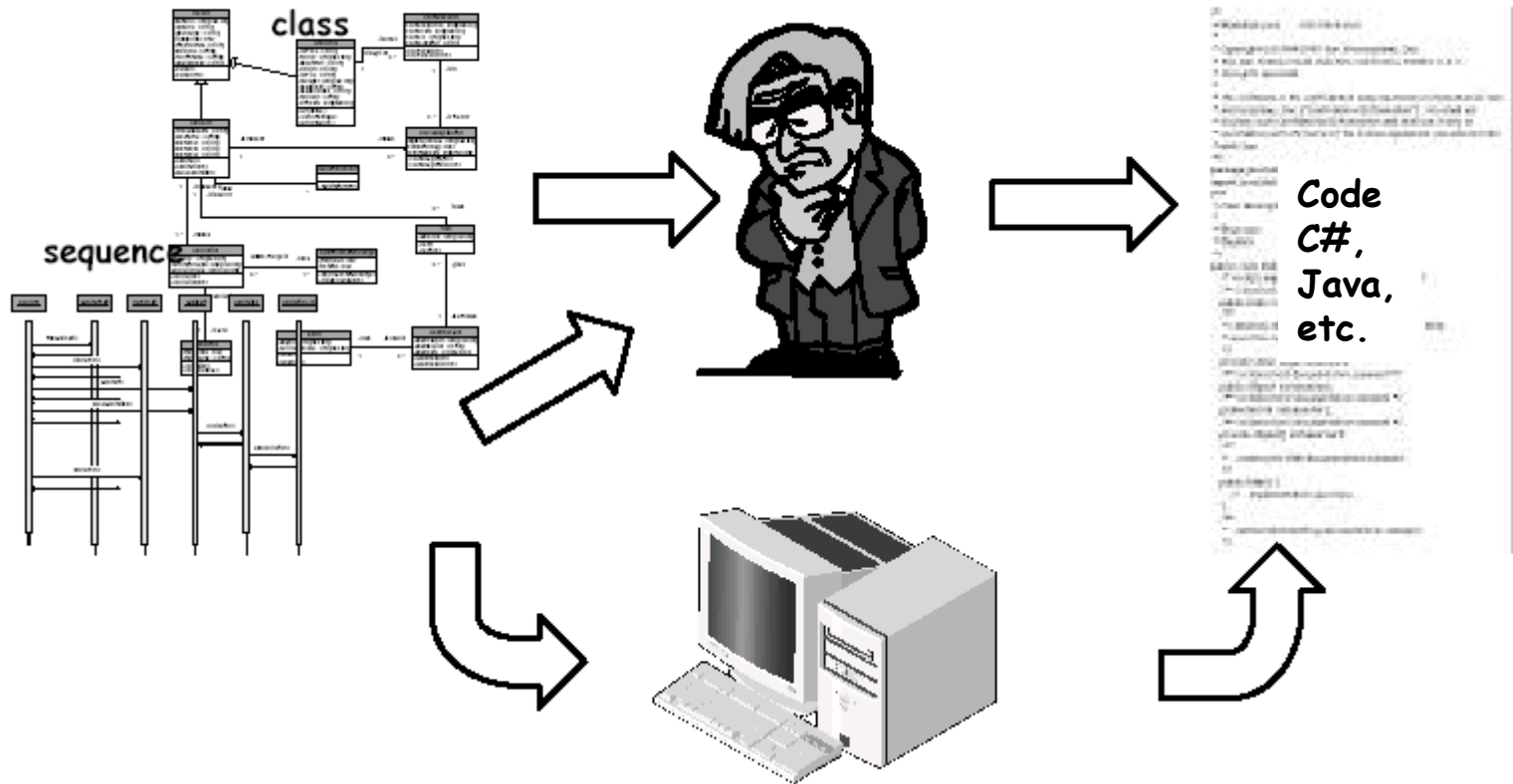
Fuente: Jean Bezivin (Keynote ICMT 2014)

- Posibles soluciones:
  - Más programadores?
  - Nuevos lenguajes de programación
  - Nuevas metodologías (agile?)
  - Mejores entornos (IDE, frameworks...)
  - Mayor reutilización (variabilidad, SPL?)
- Mejores abstracciones (lenguajes de modelado)
- Lenguajes de dominio específico
- Automatización (generación)
- End-user-programming
- ???



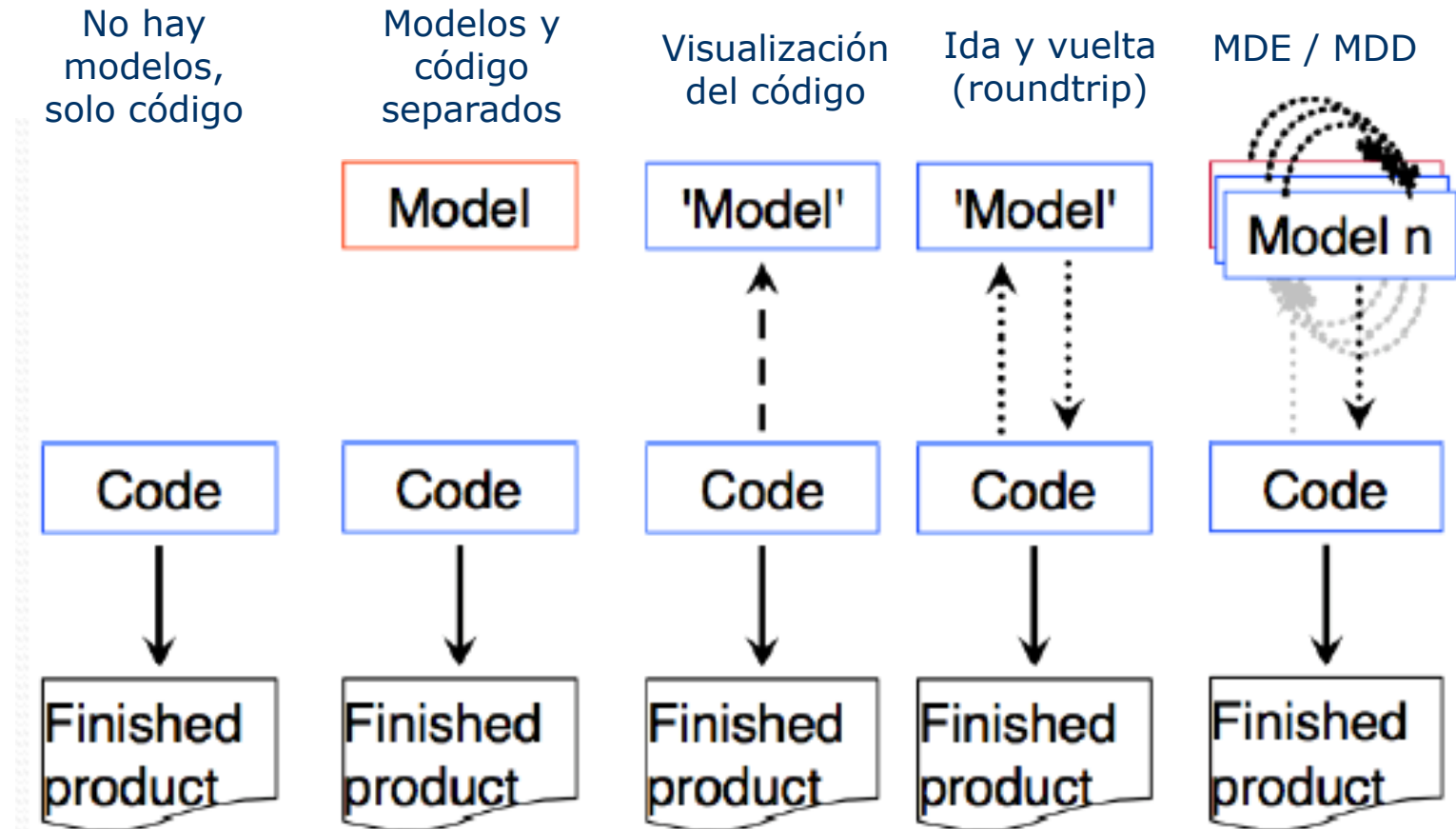
# MDD (Desarrollo de Software Dirigido por Modelos)

**MODELOS:** de artefactos *contemplativos* a artefactos *productivos*

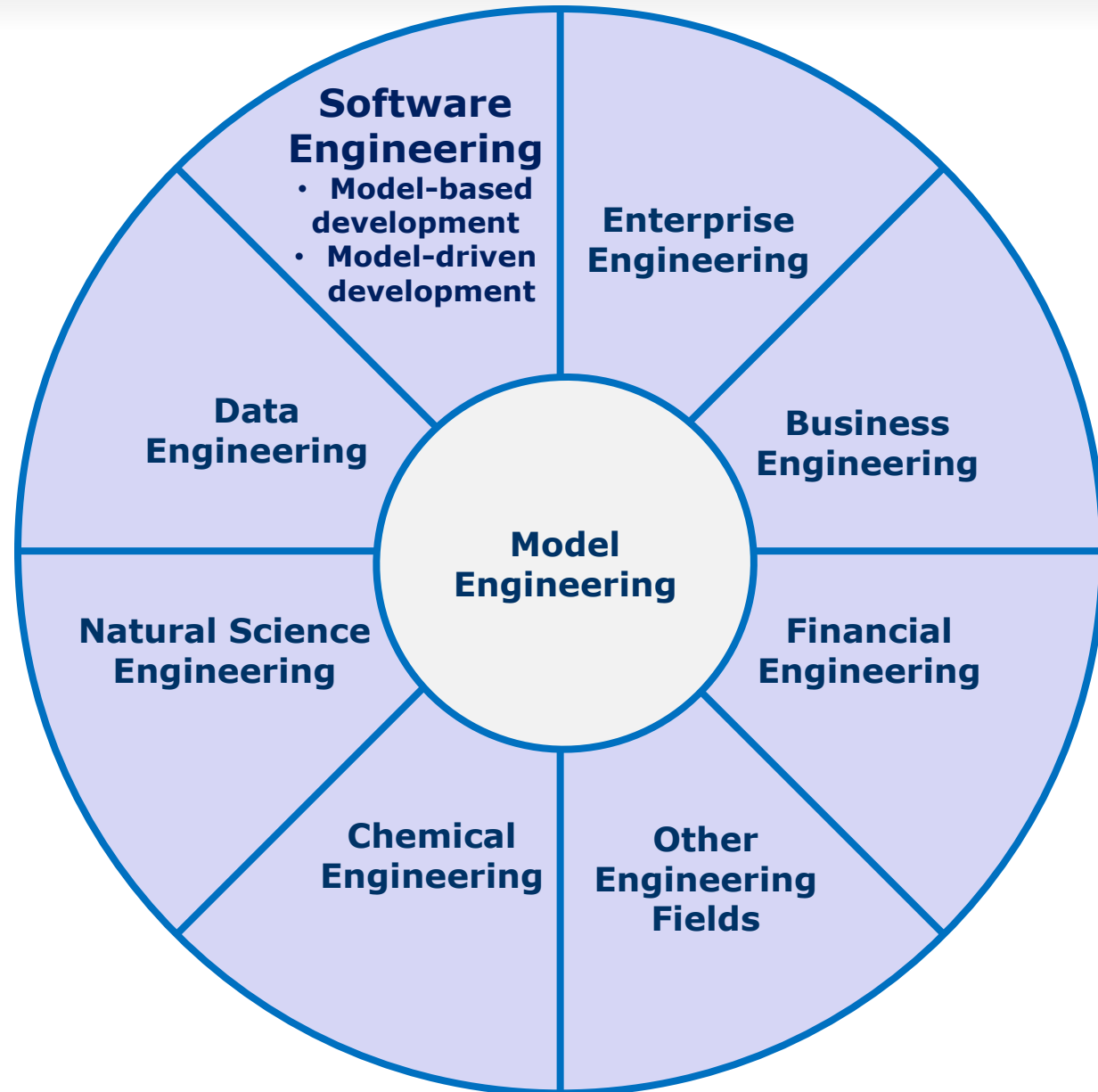


"from human-readable to computer-understandable" J. Bézivin

# Evolución del uso de Modelos en la Ingeniería del Software



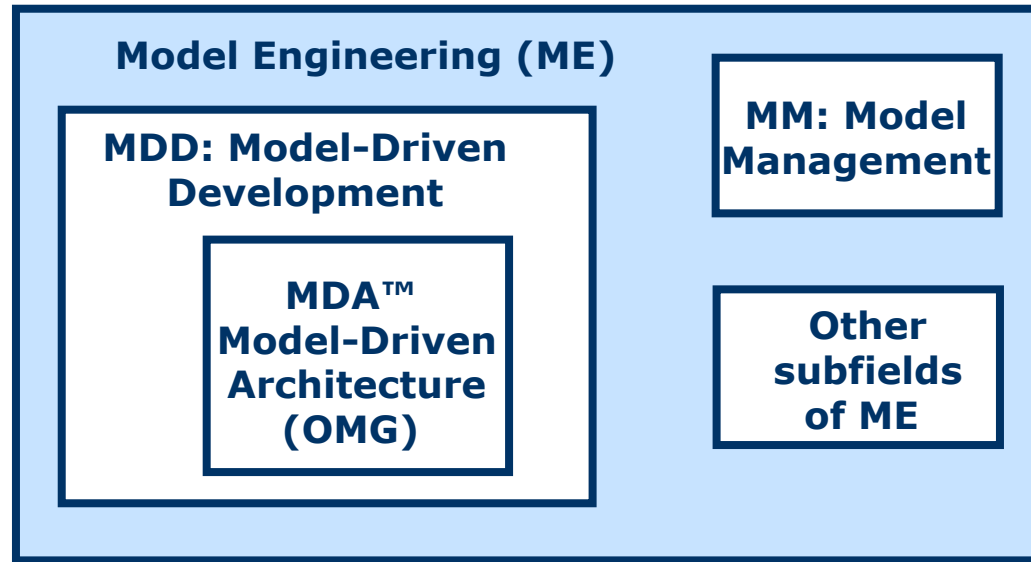
# MDE en la Ingeniería del Software



*La ingeniería de modelos es **MUY IMPORTANTE** para restringirla solo a la ingeniería del software*

*Alineamiento de los modelos del software y de la organización*

# ME, MDD y MDA™: la visión OMG



MDA™ es una forma ***específica*** de *MDD* que utiliza estándares industriales definidos por la OMG como: MOF, UML, XMI, QVT, etc.

# Object Management Group (OMG)

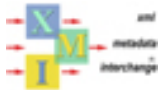
- OMG™ es un consorcio internacional de computación, abierto y sin fines de lucro desde 1989.
- El objetivo original, proveer estándares para sistemas OO distribuidos y ahora enfocado a los modelos y a los estándares basados en modelos. Algunos estándares son:



UML is a graphical language that expresses application requirements analysis and program design in a standard and methodology-independent way.



MOF (Meta Object Facility) standardizes a *meta-model* - the concepts that you use to build your application model.



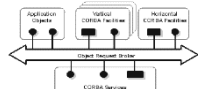
XMI is a stream format for interchange of metadata including the UML models.



CWM (Common Warehouse Model) standardizes a basis for data modeling commonality within an enterprise, across databases and data stores.



CORBA is an open, vendor-independent specification for an architecture and infrastructure that computer applications use to work together over networks

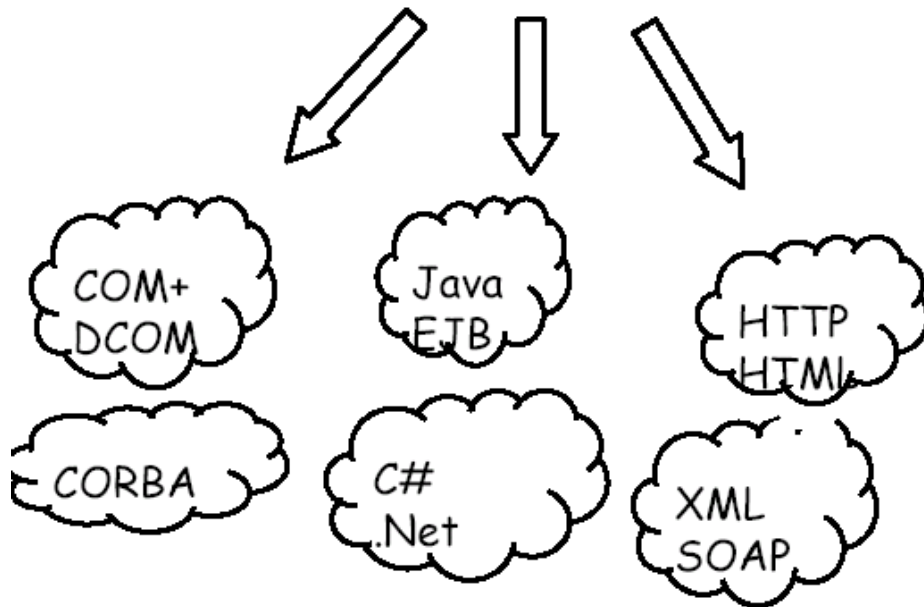


OMA (Object Management Architecture) categorizes objects into four categories: the CORBAservices™, CORBAfacilities™, CORBAdomain™ objects, and Application Objects.

# MDD/MDA: la visión de OMG

## Mapear modelos a plataformas múltiples y evolutivas

Modelos independientes de la plataforma, basados en MOF



<http://www.omg.org/mda>

- MOF/UML como base estándar.
- Valores organizacionales expresados como modelos
- Transformación de modelos para su mapeo a plataformas específicas.



# MDD/MDA: PIMs & PSMs

- **Platform Independent Model (PIM)**

“Un modelo de un sistema que no contiene información acerca de la plataforma o la tecnología que es usada para implementarlo”

- **Platform Specific Model (PSM)**

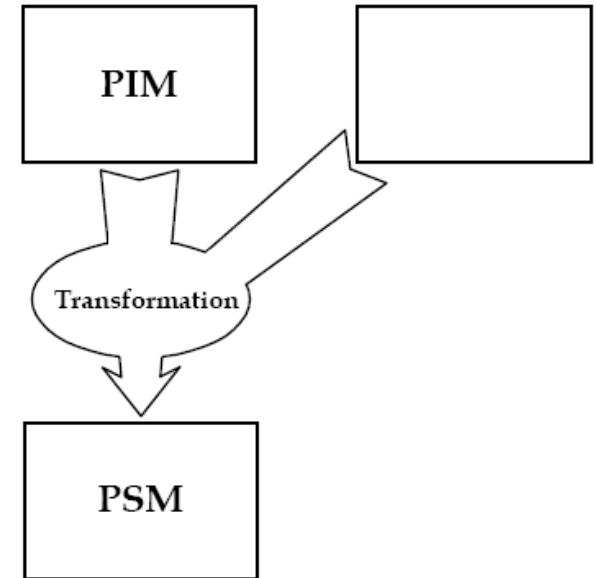
“Un modelo de un sistema que incluye información acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica”

- **Transformación de modelos**

Especifica el proceso de conversión de un modelo a otro modelo.

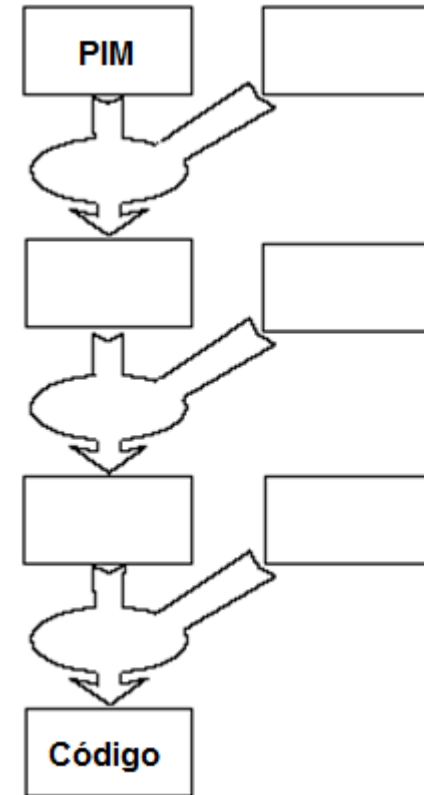
Cada transformación incluye:

- un modelo de entrada (ej., PIM)
- un modelo de salida (ej., PSM)
- una transformación
- Opcionalmente, “información adicional” que ayude a guiar el proceso de transformación (ej. Variables parametrizadas)



# MDD: aplicando transformaciones sucesivas

- El patrón MDD puede ser usado a distintos niveles para producir una sucesión de transformaciones.



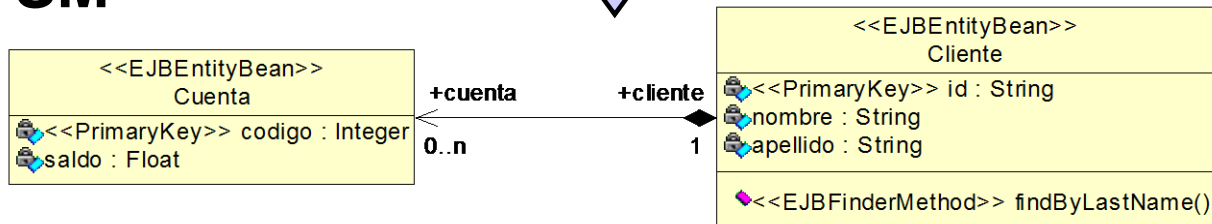


# MDD: ejemplo 1

## PIM



## PSM



## Código

```
public interface Cuenta extends EJBObject {...}
public interface CuentaHome extends EJBHome {...}
public abstract class CuentaBean
    implements EntityBean{...}
...
```

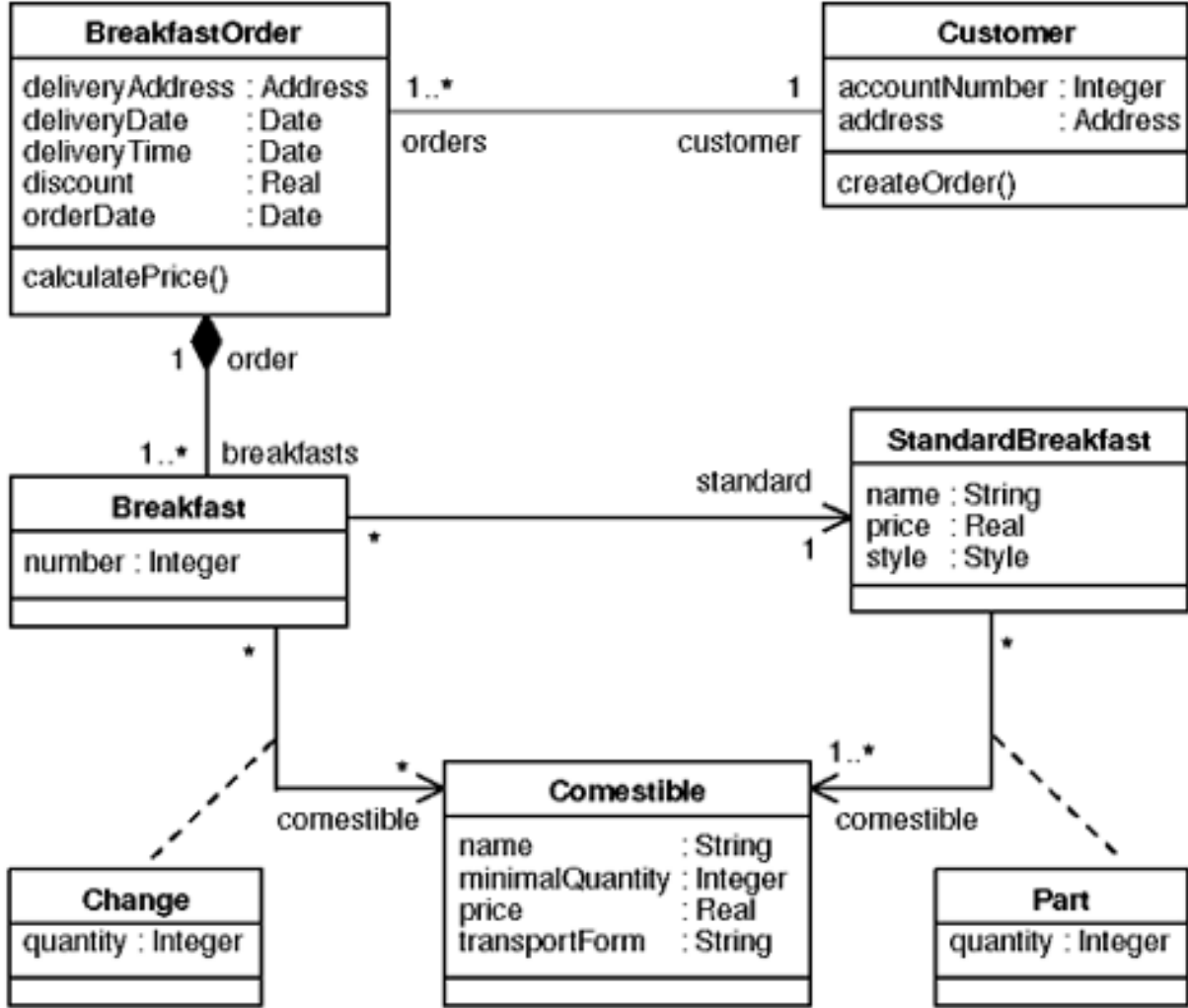
# MDD: ejemplo 2

## Servicio de Desayuno de Rosa

- Datos a ingresar: Cliente, Hora y Lugar de entrega.
- Elegir uno de los desayunos de la página web.
- Un mismo pedido puede ser para distinta cantidad de personas y distintos tipos de desayunos.
- Estilos: Simple (predefinido) o Mejorado (simple++)
- Flexibles: Es posible agregar cantidad de ingredientes, quitar otros.
- Rosa tiene 10 empleados que entran a las 5 de la mañana, 5 preparan desayunos y 5 los entregan.



# PIM en detaille



# Capas de la Aplicación

## Interfaz de Usuario

Java Server Pages  
(JSP)

## Capa de Negocio

Enterprise Java Beans  
(EJB)

## Base de Datos

Relacional

# MDD – del PIM al PSM

- PIM conceptos *sin* nivel tecnológico
- PSM son *dependientes* de la plataforma
- CODIGO *específico* para la plataforma

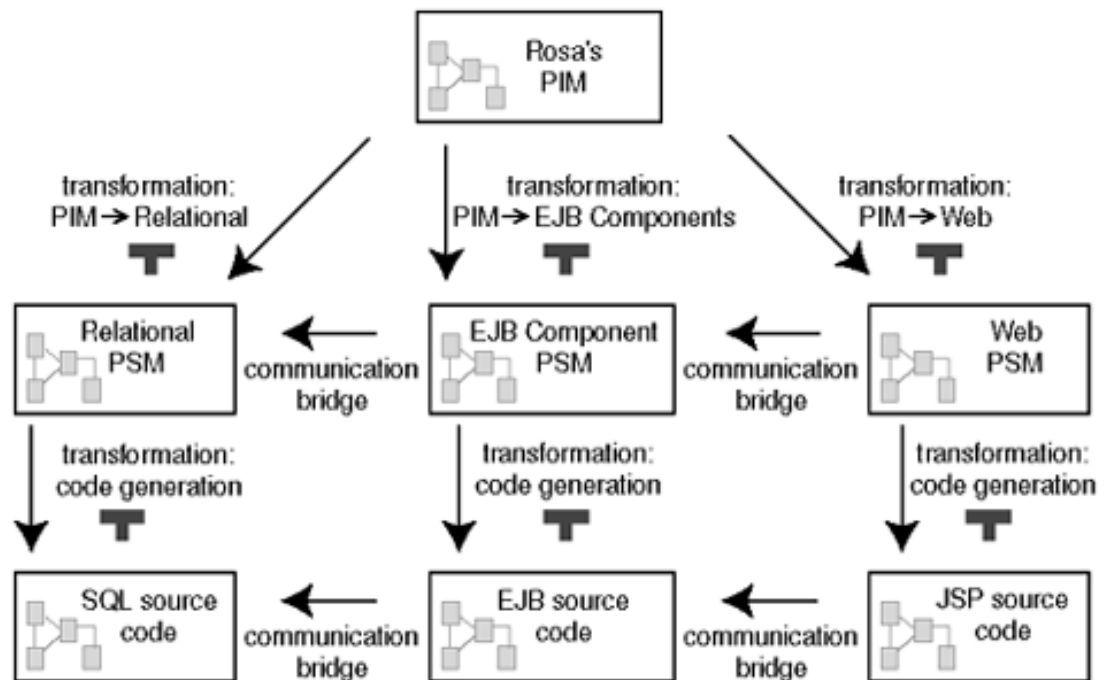
Como cada capa usa una tecnología diferente se crean 3 PSM uno para cada capa.

Capa de Base de Datos: el modelo será un DER

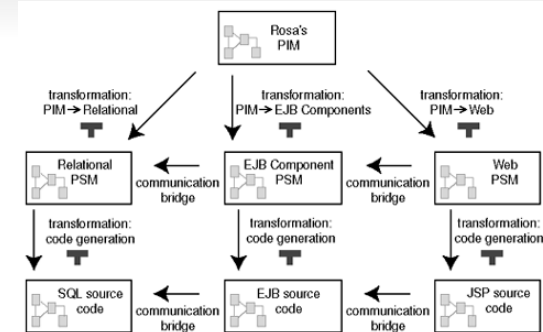
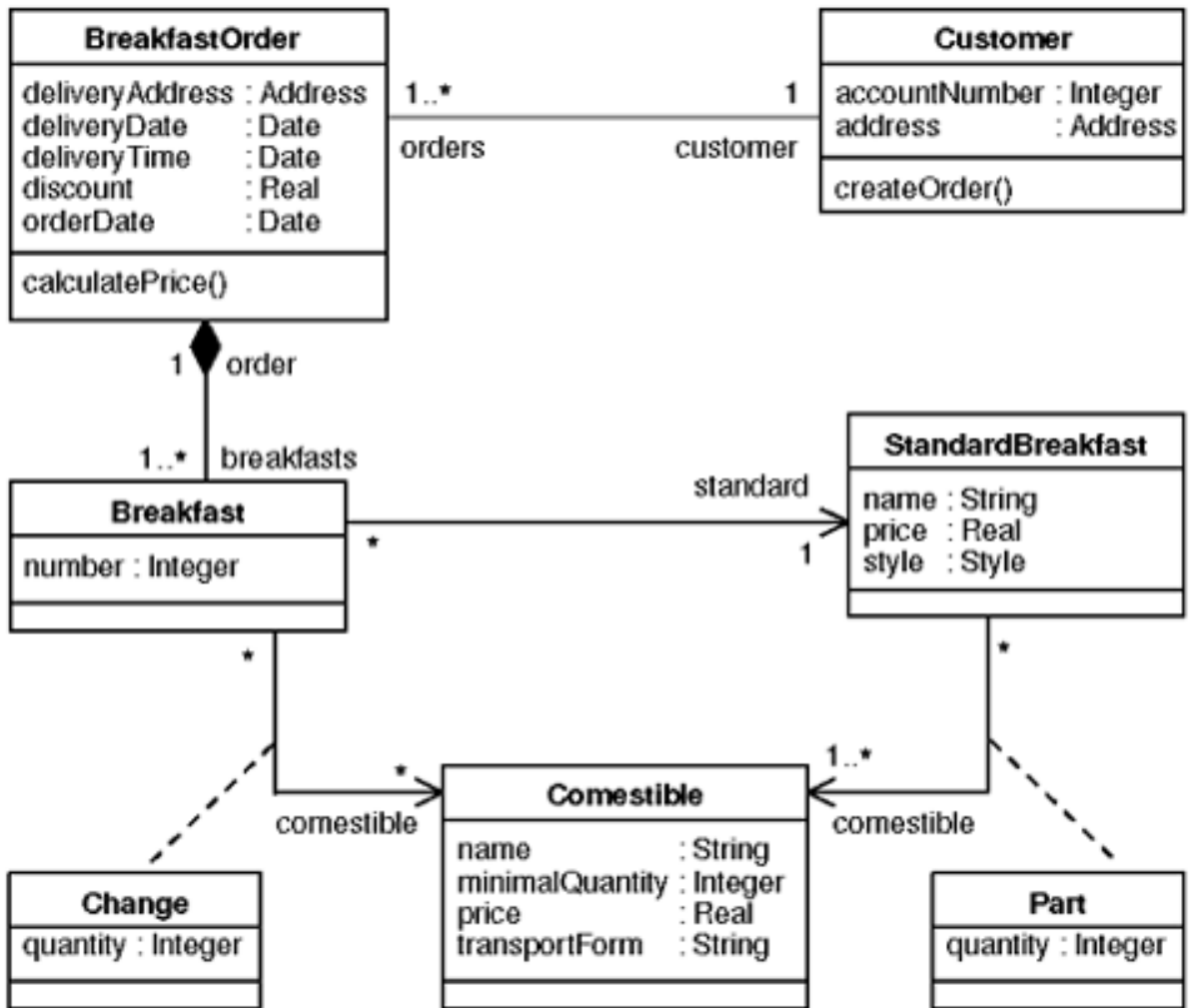
Capa de Negocio: Se crean modelos de UML con estereotipos para el EJB

Capa de interfaz: se crean modelos UML con estereotipos para la WEB

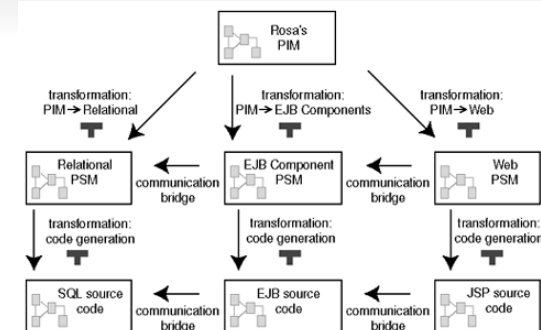
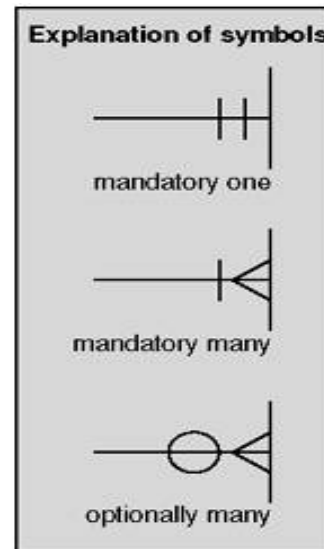
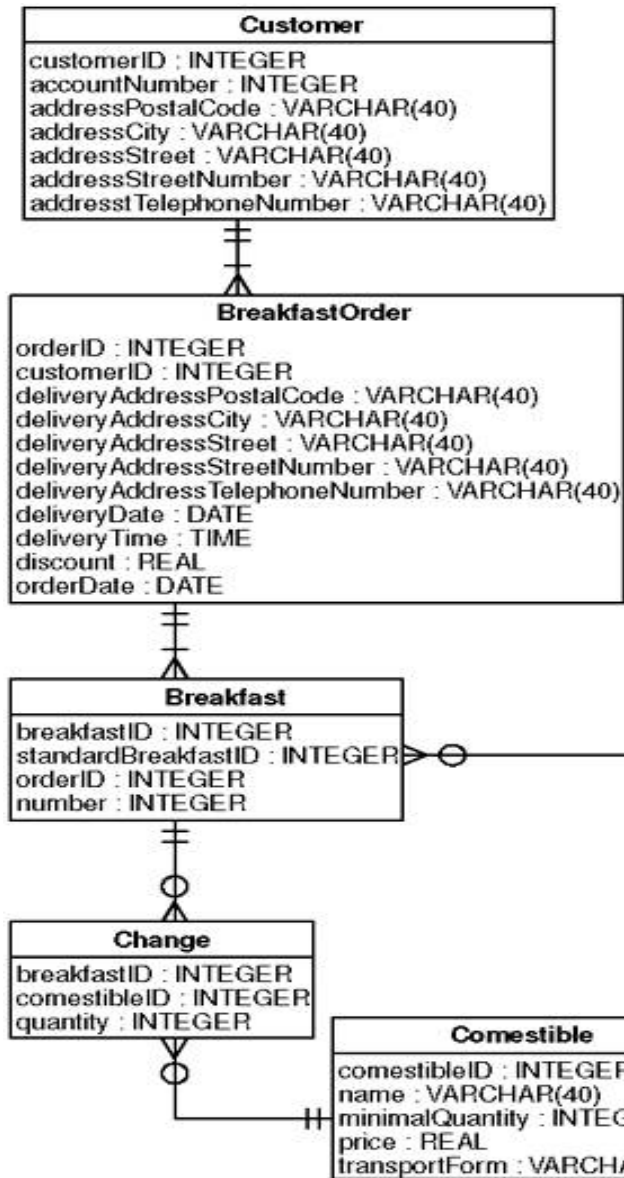
**Modelos interoperables!!**



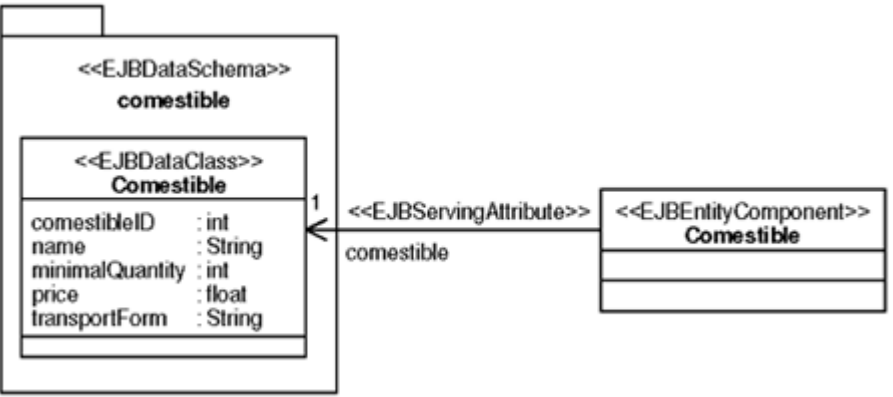
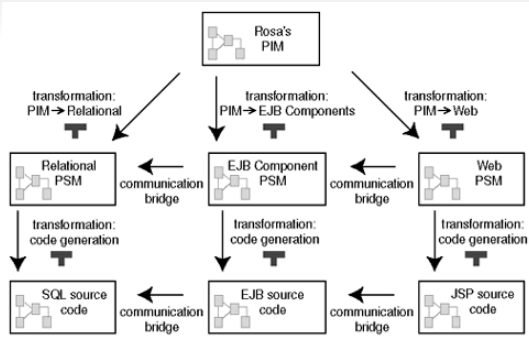
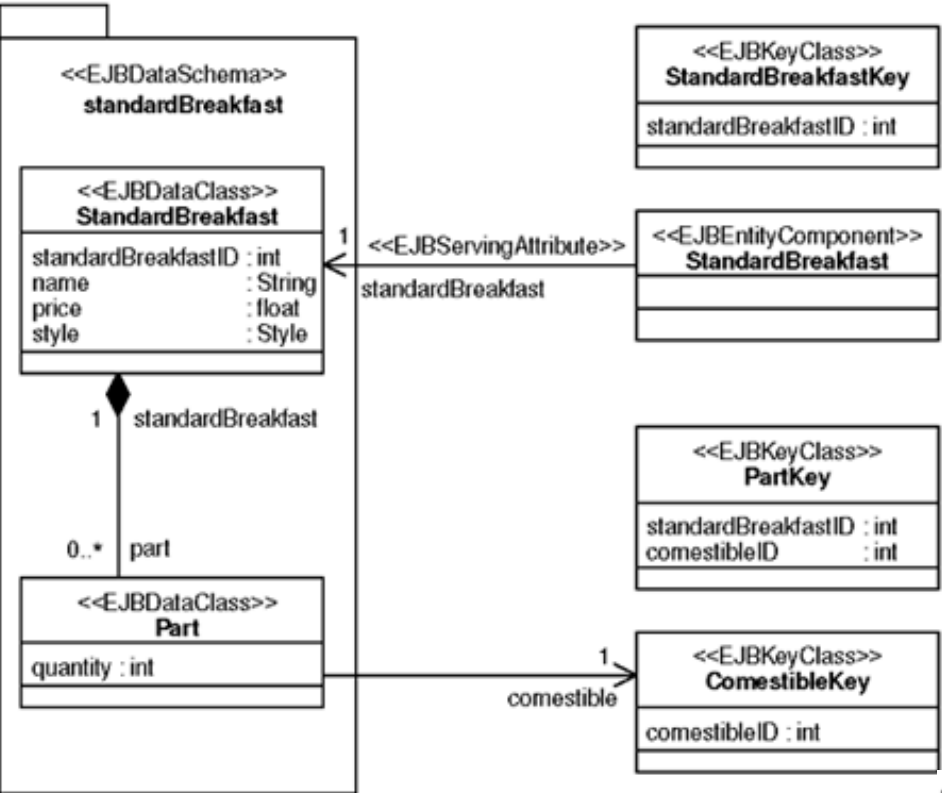
# PIM detallado



# PSM relational

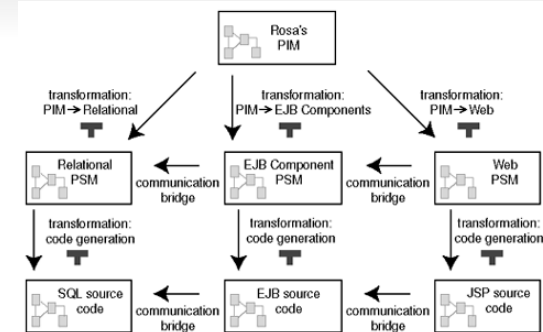
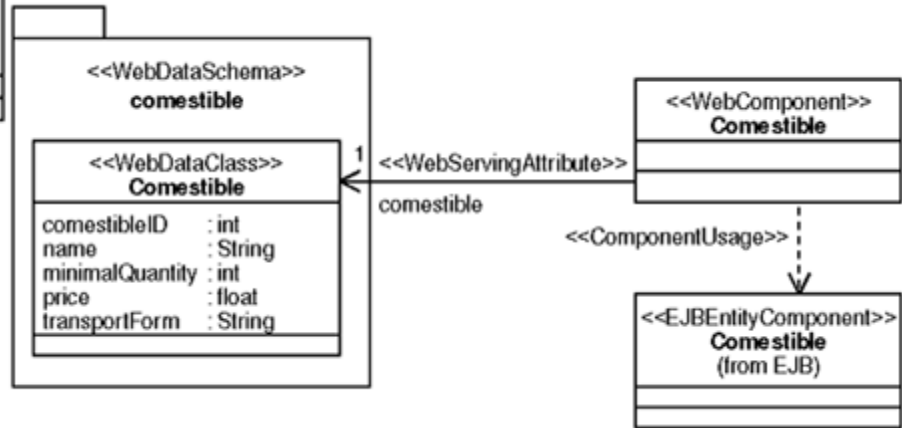
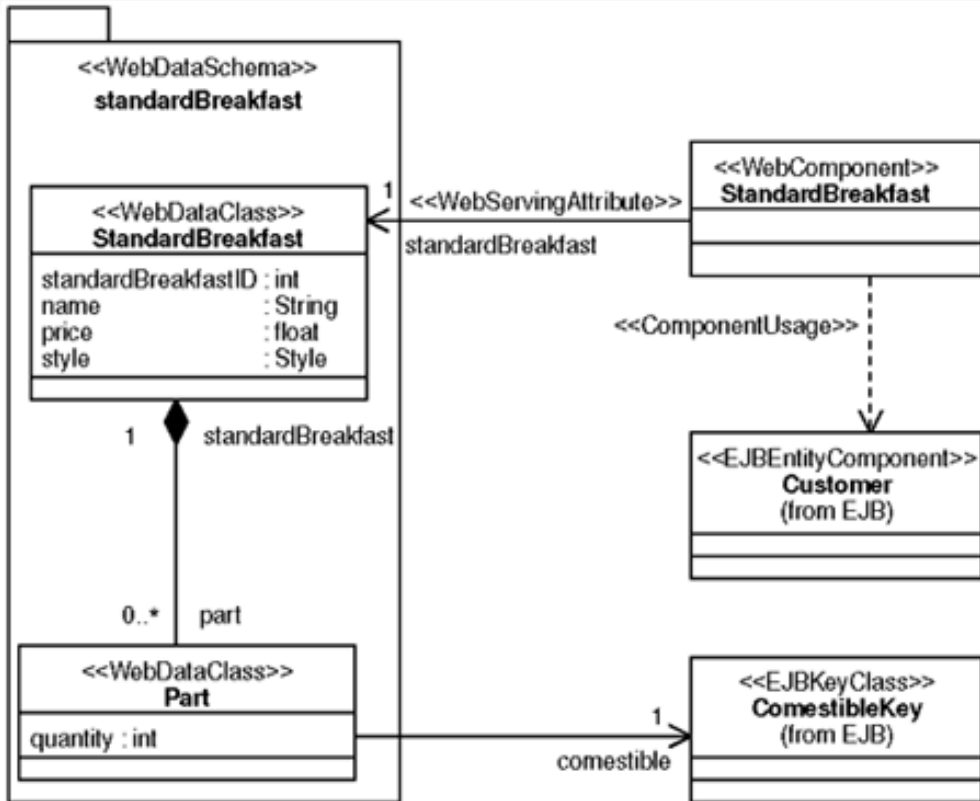


# PSM para EJB

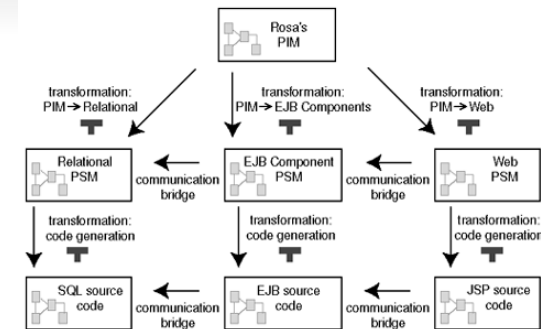
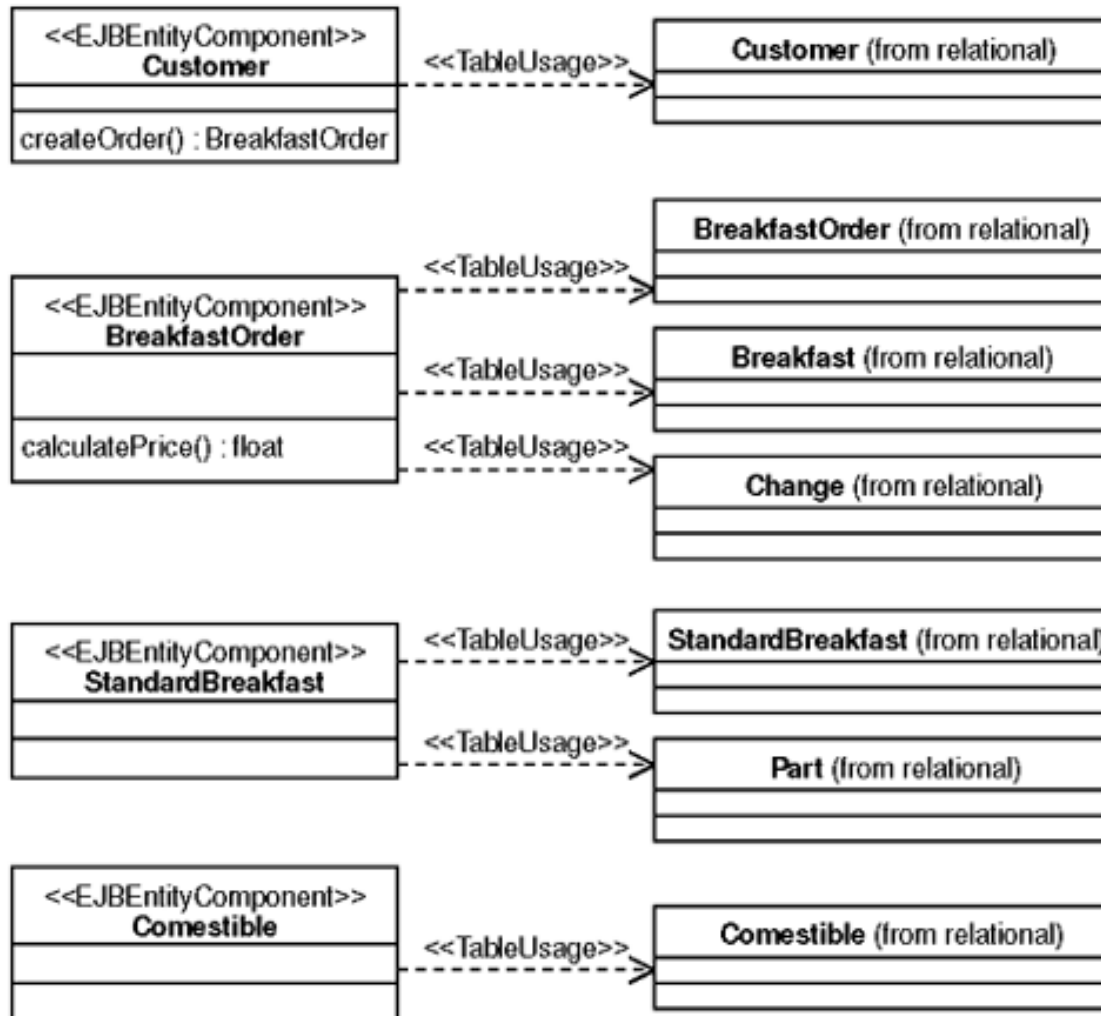




# PSM para aplicación Web



# Puentes de Comunicación: Capa de Negocio – Capa de Datos



## **II. MOF (Meta-Object Facility) y Metamodelado**

<http://www.omg.org/mof/>

**MOF 2.4.2  
ISO/IEC 19508:2014 Tecnología de  
información — Meta Object Facility (MOF)**

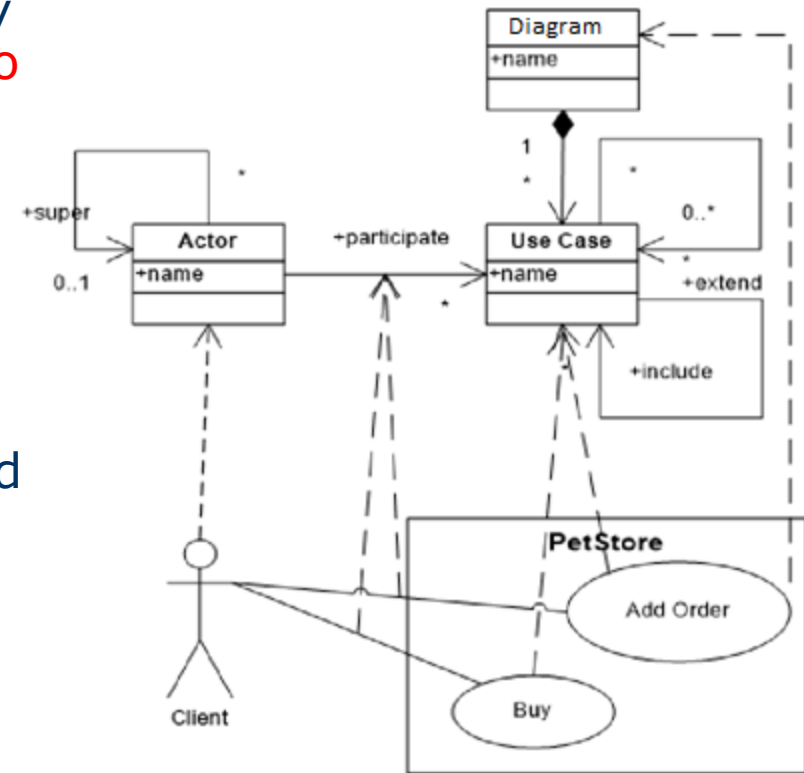
**Última versión: 2.5.1 (Noviembre de 2016)**

# Metamodelado

- **Metamodelado** denota la definición de **modelos para modelos**
- Analogías:
  - Si un modelo es un language, su metamodelo es la **sintaxis** y **semántica** (estática) de dicho lenguaje
  - Si el modelo es considerado un objeto, el metamodelo puede ser visto como el **metaobjeto** que lo ha creado
- ¿Por qué metamodelar?
  - Los modelos no son **ambiguos** si son definidos formalmente (sintaxis y semántica bien definidos)
  - La **interoperabilidad** y **manipulación** solo es posible si están basados en un metamodelo (conocido y bien definido)
- Los metamodelos son también modelos
  - Se usa (un subconjunto de) UML para definir los metamodelos
- Además, el propio UML está definido en términos de un metamodelo llamado MOF (Meta Object Facility).
  - También MOF usa UML para su representación

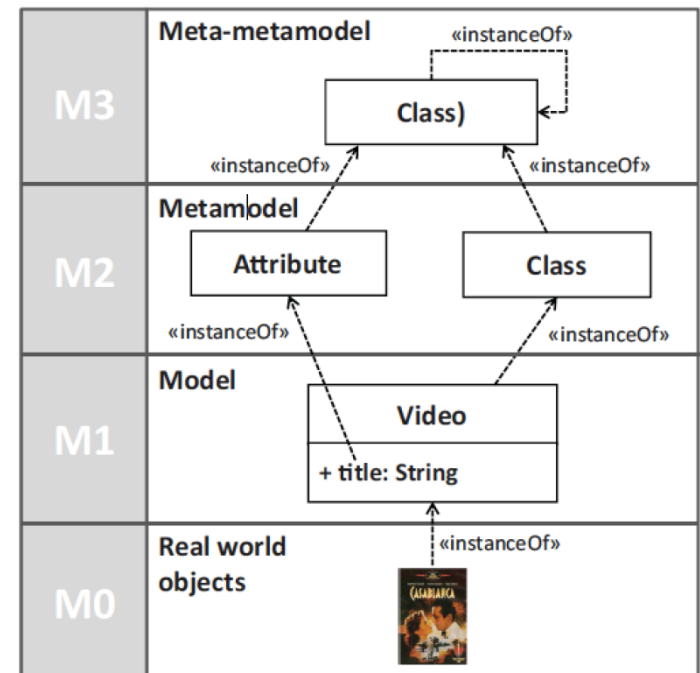
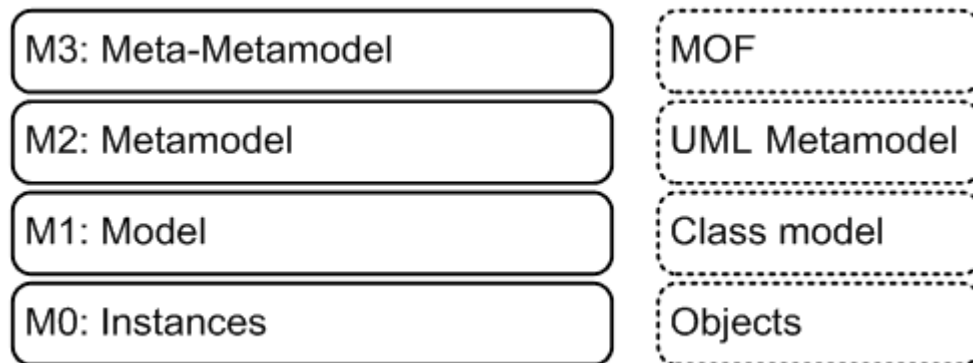
# Metamodelos

- Un metamodelo define los conceptos y sus relaciones usando un (subconjunto del diagrama de clases UML)
- Un metamodelo solo define la estructura (y la semántica estática)
- Un modelo es “conforme” a un metamodelo **sii** respeta la estructura d su metamodelo
  - ej. el metamodelo Casos de Uso UML define la estructura de todos los modelos de Casos de Uso!



# MOF - Una Arquitectura de Metamodelos

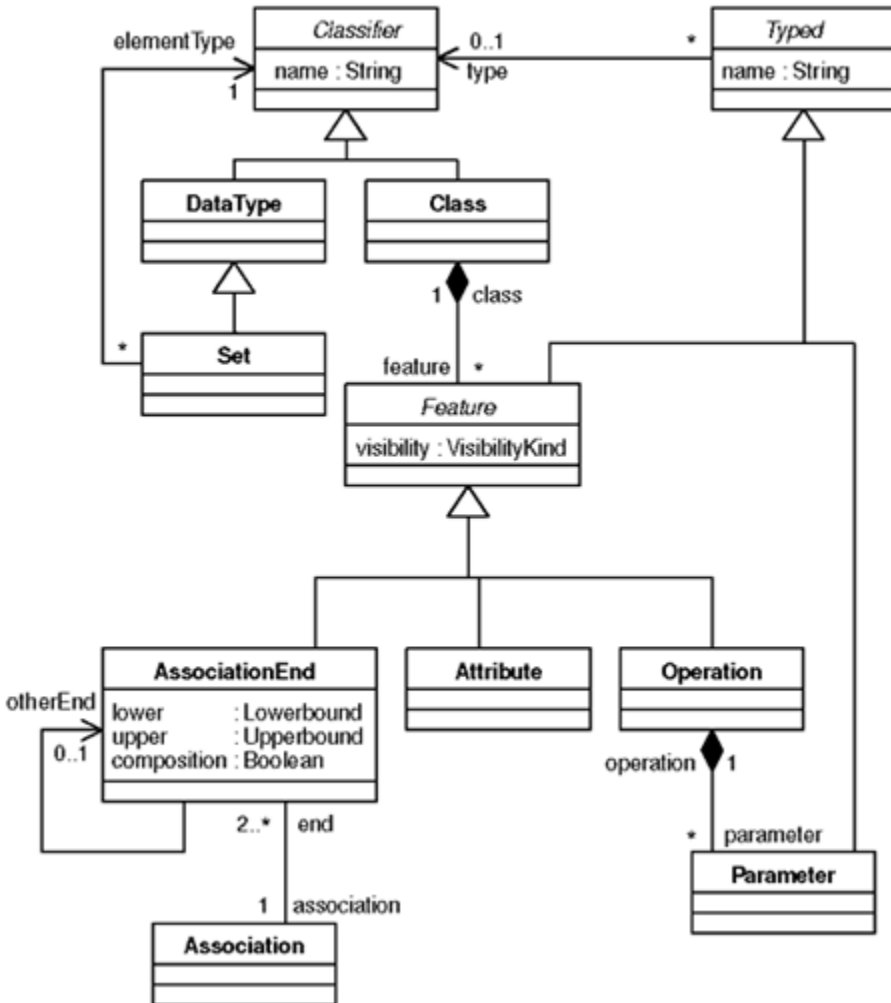
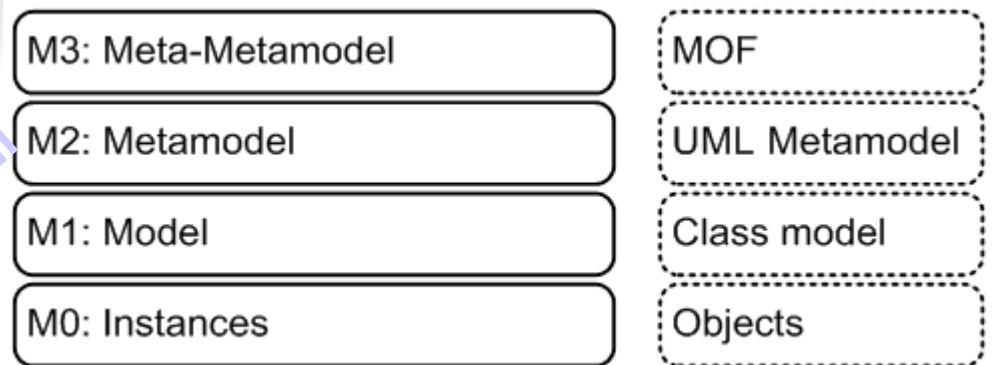
- El Meta Object Facility (MOF) provee un framework de gestión de metadatos para permitir el *desarrollo* y la *interoperabilidad* de sistemas dirigidos por modelos y metadatos.
- MOF reside en el nivel M3 de una arquitectura de metamodelos de cuatro niveles



- **M0** objetos
- **M1** es específico a un **sistema software**
- **M2** es específico a un **modelado OO (UML)**
- **M3** es **no específico (genérico)**

# MOF y Metamodelado

**Metamodelado** determina la definición de modelos para los modelos



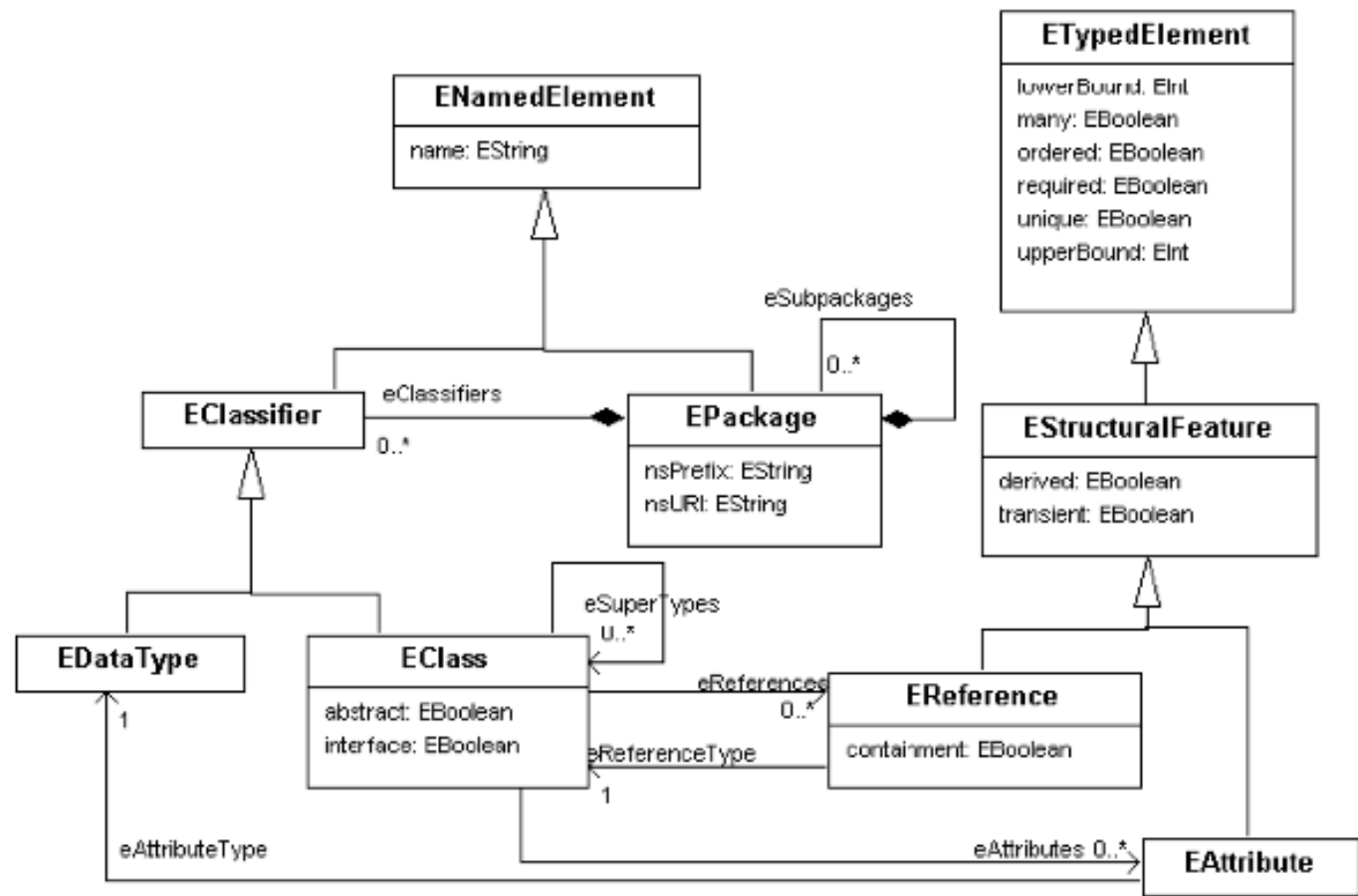
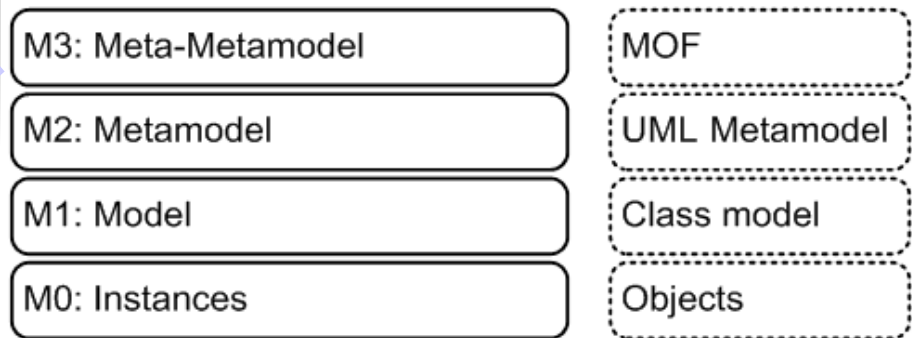
- **UML** es usado para definir modelos (**M1**)
- **UML** está definido **por un metamodelo (M2)**
- El metamodelo de UML está definido **por un meta-metamodelo** llamado **MOF (M3)**

# MOF (Nivel M3) - Detalles

- Elementos principales de MOF (*conceptos para definir un lenguaje*):
  - **Clases**, que modelan metaobjetos MOF (ej. Método)
  - **TiposDeDatos (propiedades)**, que modelan datos descriptivos de los metaobjetos (ej., nombre del Método)
  - **Asociaciones (propiedades)**, que modelan relaciones binarias entre metaobjetos (ej. el Método *tiene* Parámetros)
  - **Paquetes**, que modularizan el modelo
- OMG define 2 variantes de MOF:
  - EMOF → *Essential* MOF
  - CMOF → *Complete* MOF
- La variante **ecore** definida en el **Eclipse Modeling Framework (EMF)** se basa en el **EMOF**



# MOF (Nivel M3) - Detalles



# Diferencias entre el metamodelo UML y el metamodelo MOF

Las diferencias se deben al **uso y propósito diferente** de UML y MOF

- Modelar con MOF no es modelar con UML
- MOF soporta solo *relaciones binarias* mientras que UML relaciones *n-arias*
- MOF no soporta las clases asociación o asociaciones cualificadas de UML
- MOF soporta el concepto de **referencia** para permitir la “navegación” de un elemento a otro. UML usa el concepto de roles de asociación para este propósito
- Algunos conceptos como la **Herencia** son representados en el metamodelo de UML como metaclases, pero en MOF son **referencias**
- ...

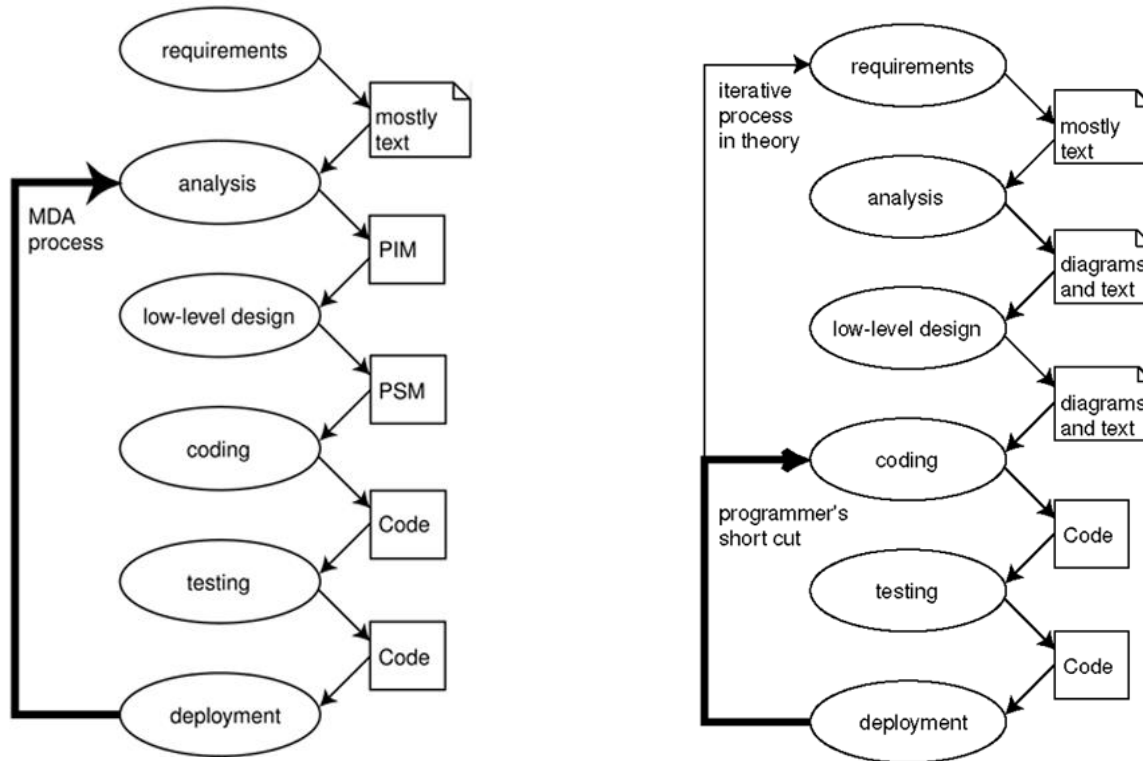
# MOF Conclusiones

- MOF es una arquitectura de metamodelado *cerrada*  
(todos sus elementos, a distintos niveles, están definidos dentro de la misma arquitectura)
- MOF permite una arquitectura de matamodelado *estricta*  
(cada elemento del modelo en cada nivel se corresponde (*conforms*) a un elemento del modelo del nivel superior)
- MOF define la *sintaxis abstracta* de un lenguaje de modelado  
(identifica los elementos constituyentes del lenguaje y cómo se pueden relacionar entre ellos para construir un modelo sintácticamente correcto)
- Para la definición de metamodelos, MOF desempeña el mismo rol de las BNF para definir las gramáticas de los lenguajes de programación
  - Backus-Naur form (BNF) es un metalenguaje usado para expresar gramáticas libres de contexto  
ej.: <dirección postal> ::= <nombre> <dirección> <código postal>
  - MOF se usa para definir metamodelos así como las BNF se usan para definir gramáticas

# **III. Transformaciones**

# Transformación de Modelos en MDD

- El proceso MDA vs. el desarrollo tradicional

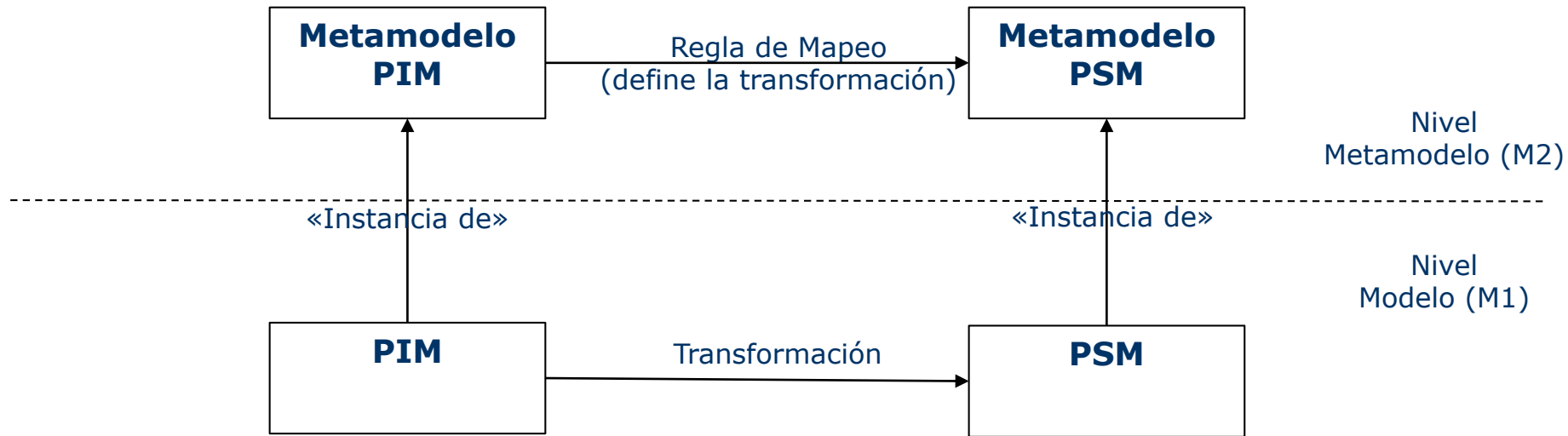


- En MDD, las transformaciones son ejecutadas por herramientas
  - Lo realmente **nuevo** en MDD es que las transformaciones entre modelos (ej., PIM a PSM) son **automatizadas**



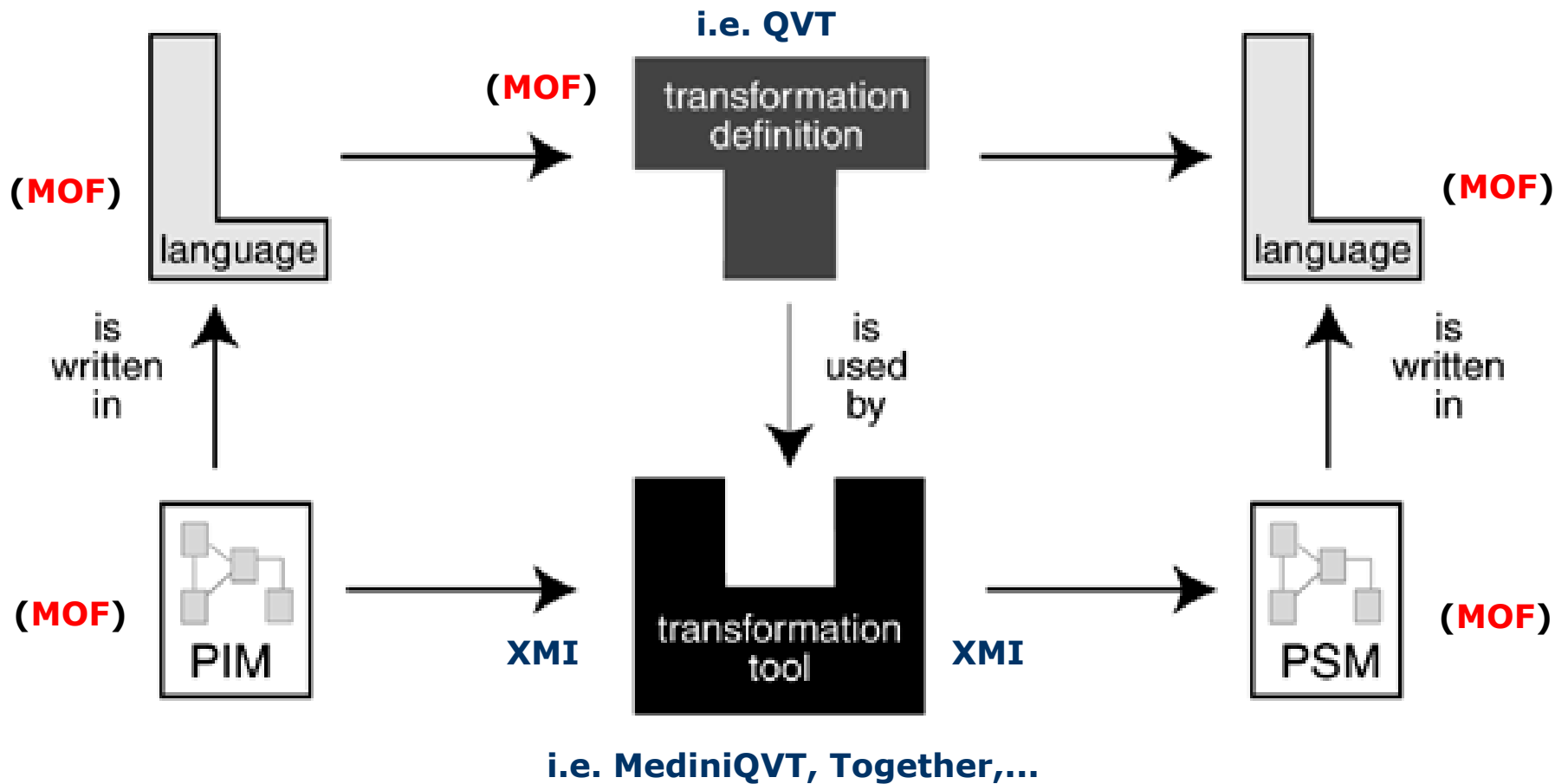
# Transformación de Modelos en MDD

- **PIMs** son independientes de las plataformas de implementación
  - Solo capturan la lógica del negocio – el **espacio del problema**



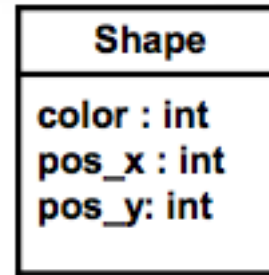
- Un conjunto de **transformaciones** de modelos se aplican para convertir el modelo **independiente** de plataforma en **específico** de una plataforma
- Una transformación de modelos se **define** con una **Regla de Mapeo** al *nivel de metamodelo* pero se **ejecuta** a *nivel de modelos*!

# Transformación de Modelos en MDD

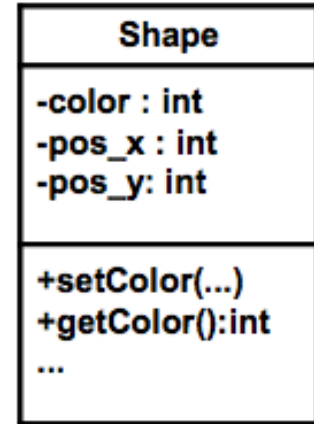


# Transformaciones de Modelos: Ejemplo

- Una clase *shape* en un **modelo de análisis** (PIM) →



- Una clase *shape* en un **modelo de diseño** (PSM) →
- Transformación** (en pseudocódigo):



```
foreach class:a in AnalysisModel {  
  define class:d in DesignModel;  
  
  foreach a.attribute:attr {  
    define operation:setter on d with setter.name =  
      "set"+attr.name, setter.type="void" .... ;  
  
    define operation:getter on d with getter.name =  
      "get"+attr.name,getter.type=attr.type  
  }  
}
```



# Usos de Transformaciones en Modelos en MDD

- **PIM → PIM:** mapeo cuando no es necesaria información específica de plataforma Ej.: especialización en un dominio, análisis a diseño
- **PIM → PSM:** el PIM se transforma a un modelo específico de plataforma
  - Mapeo basado en las características de una plataforma concreta. Habitualmente modelado usando perfiles específicos de UML. Ej.: modelo de análisis a un modelo de diseño)
- **PSM → PSM:** cuando el PSM se debe ajustar una versión o características específicas de implantación. Ej.: SQL ANSI87 a SQL 2005 o SQL 2008
- **PSM → PIM:** para obtener PIMs a partir de PSMs concretos
  - Reingeniería
  - Difícilmente automatizado en su totalidad
- **Generación de código:** PSM→código o PIM→código
- **Compilación de modelos:** PIM→PSM→código (transformación sucesiva de modelos hasta llegar al código, habitualmente el encadenamiento M2M y M2C).

Obs: algunos autores usan el término *generación de código* como sinónimo de *compilación de modelos*.

# Conclusiones I

- Las técnicas de Ingeniería de Modelos están basadas en:
  - Una arquitectura de Modelos, Metamodelos, ... (MOF)
  - Mecanismos uniformes de transferencia e intercambio (XMI)
  - Mecanismos de transformación entre modelos (ATL, QVT,...)
  - Mecanismos de generación de código: Java, .NET, ... (MOF2Text)

# Conclusiones II

- MDD no es una nueva tecnología, pero...
  - Una forma de tratar nuevas tecnologías emergentes: servicios web, dispositivos móviles,... de una forma natural
  - Integración progresiva de:
    - Aplicaciones legadas (Cobol, RPG, ADA, PL/1, Pascal, etc.),
    - Aplicaciones actuales (Java, CORBA, Web services, etc)
    - Aplicaciones futuras: cloud computing, grid computing,...y todo en un contexto donde los sistemas se pueden adaptar y evolucionar con las necesidades y las tecnologías emergentes
- **MDD** no es solo desarrollo, es sobre todo integración y evolución

# Conclusiones III

- Además, permite tratar la creciente **complejidad** de los nuevos sistemas software
- La fuente de **complejidad** es múltiple:
  - *Volumen (de datos, de código...)*
  - *Evolución (tecnológica, del negocio,...)*
  - *Heterogeneidad*
    - SO y middlewares
    - Lenguajes
    - Paradigmas (procedimientos, eventos, objetos, servicios, reglas, aspectos, etc.)
    - Redes, Bases de Datos, etc.

MDD *gestiona la complejidad por medio de la **abstracción** y la **automatización***

## Conclusiones IV

- MDD probablemente NO es la solución a todos los problemas de la Ingeniería del Software,

pero:

- por ahora, no hay una alternativa mejor
- el siguiente paso “lógico” de madurez (*artistas vs ingenieros*)
- intenta **desacoplar** la parte del **negocio** de la parte **tecnológica** en los sistemas, que ha sido un desafío histórico en la ingeniería del software.

## Conclusiones V (herramientas)



django

**Siguen los principios  
básicos de MDD!!!!**



RAILS



**Dado un modelo, crean  
la base de datos y toda  
la interfaz para CRUD**

**GeneXus™**

Grow thru knowledge

Fuente: <http://modeling-languages.com>

No quiero volver a pagar un alto precio para "simplemente" migrar nuestro sistema de información a una nueva plataforma (Java, HTML, XML, DotNet, Cloud, etc.) cuando nuestro **modelo de negocio es el mismo!!!**

Quiero poder pagar ese precio por definir nuestros modelos y servicios de negocio de forma que nos **garantice** que **no** vamos a quedar tecnológicamente obsoletos.

Así, en el futuro, cualquier proveedor de nuevas plataformas nos podría vender soluciones de "mapeo" desde nuestros modelos y servicios de negocio a esta nueva plataforma.

