

# MDA: Arquitectura Dirigida por Modelos

Uno de los principios básicos de la ingeniería de software es la abstracción, para separar lo esencial de lo no esencial.

En términos de negocio, lo esencial es la funcionalidad, y lo no esencial la plataforma tecnológica.

Estas abstracciones nos las proveen los modelos.

El modelado y la transformación de modelos, hasta el nivel de abstracción requerido, constituye el núcleo del desarrollo dirigido por modelos.

Los modelos se utilizan para capturar los requisitos, y automatizar total o parcialmente el desarrollo.

Con los modelos podemos centrarnos en el diseño lógico de la aplicación, y liberarnos de los detalles de la implementación.

El esfuerzo invertido en el modelado tiene una continuidad durante el desarrollo, estos modelos no son meramente parte de la documentación sino que dirigen de forma automatizada el desarrollo de código.

Model Driven Architecture (MDA) es un framework para el desarrollo de software definido por el Object Management Group (OMG)

MDA es una vía para organizar y gestionar las arquitecturas empresariales soportada por herramientas automatizadas para definir los modelos y facilitar las transformaciones entre ellos.

Con MDA el proceso de desarrollo está dirigido por la actividad de modelar el software. Utiliza estándares abiertos de modelado, como Unified Modeling Language (UML)

Modelos independientes de la plataforma tecnológica pueden ser transformados en plataformas empresariales, libres o propietarias, incluyendo J2EE, .Net, PHP ....

Tenemos 3 tipos de modelos en función del nivel de abstracción:

- CIM (Computation Independent Model)
  - Modelado de negocio y requerimientos
- PIM (Platform Independent Model)
  - Análisis y diseño independiente de la plataforma tecnológica
  - El analista realiza el PIM que describe el sistema
  - Un PIM se transforma en uno o mas PSM
- PSM (Platform Dependent Model)
  - Diseño dependiente de la plataforma tecnológica
  - El arquitecto adapta el modelo para una implementación tecnológica específica
  - Cada PSM se transforma en código

# Transformaciones

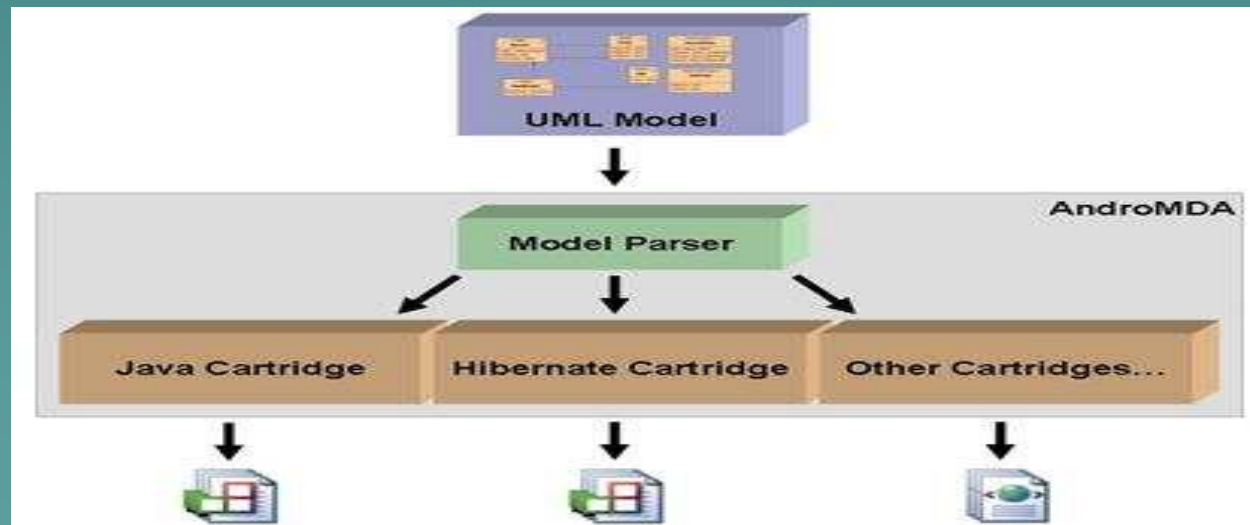
- La transformación de modelos es una parte clave de MDA.
- La herramienta de transformación coge un PIM y lo transforma en uno o mas PSM.
- Una segunda (o la misma herramienta) transforma el PSM a código.
- Poder adaptar las transformaciones a nuestras necesidades es uno de los puntos fuertes de MDA

# AndroMDA

- <http://www.andromda.org>
- AndroMDA es un framework MDA open source, recibe como entrada un modelo UML en formato XMI, los combina con los plugins de AndroMDA (cartridge y translation libraries) y produce código fuente.
- El proceso de transformación se controla utilizando los llamados cartridges. Pueden modificarse para adaptarlo a nuestras necesidades.
- AndroMDA significa principalmente escribir menos código manualmente, eliminando tareas repetitivas.

# AndroMDA

- Cuando necesitas modificar la aplicación, se cambia primero el modelo, se regenera el código, y se añade o modifica el código personalizado.
- La generación de código viene gobernada por los cartridges, que pueden ser personalizados. Existen cartuchos para Java, .Net, BPM4Struts, jBPM, JSF, EJB, Hibernate, Spring, WebService ...



## Modelado UML

- Toma de requisitos
  - Diagrama de casos de uso
- Diagramas de clases
  - Diagrama de dominio
  - Diagrama de Value Objects
  - Diagrama de servicios
- Procesos
  - Diagramas de actividad

## UML Profiles

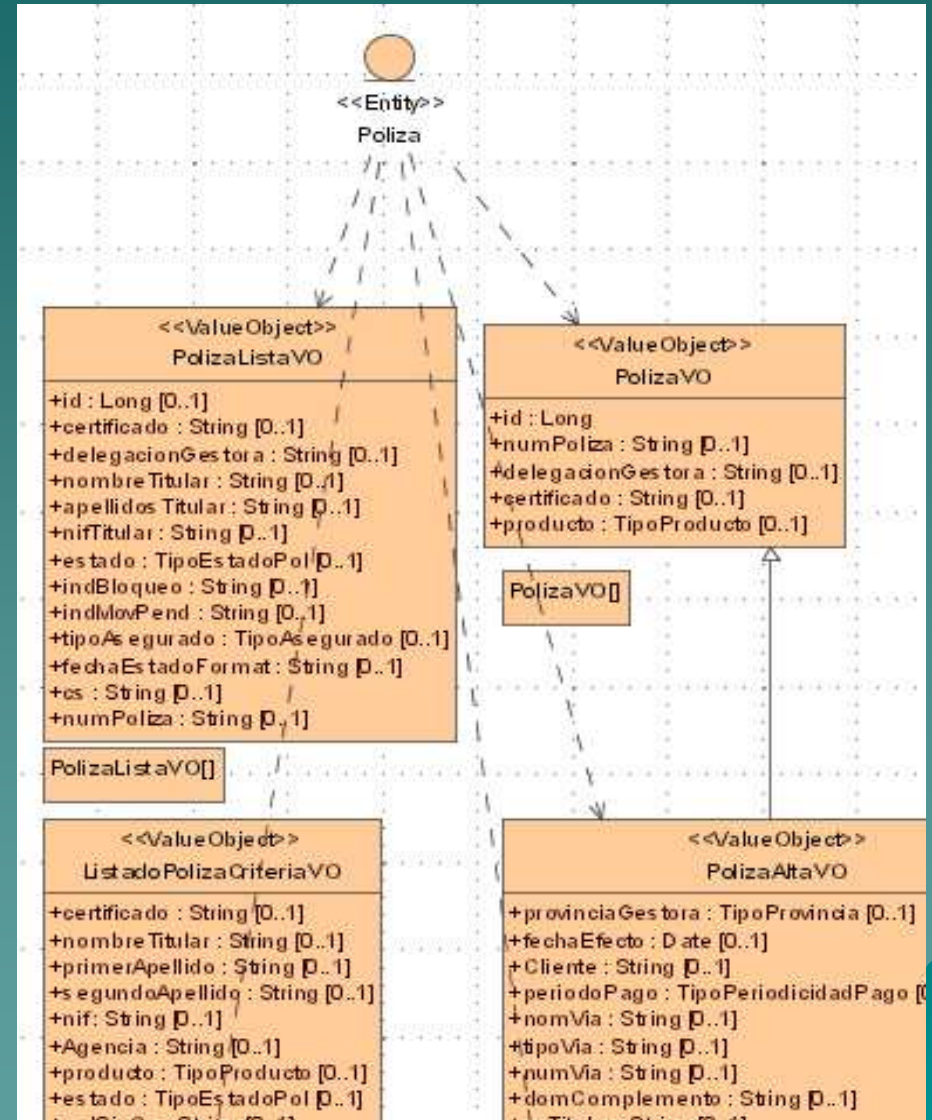
- Son extensiones para adaptar UML a un dominio particular. Introduce un conjunto de stereotypes que extienden UML.
- Los estereotipos son convertidos en código. La lógica de negocio debe ser implementada manualmente.
- MagicDraw como herramienta recomendada para modelado. Eclipse para desarrollo.

# VALUE OBJECTS

- Clases que encapsulan información para el usuario final. Representan diferentes vistas sobre los objetos de negocio.
- Recogen información procedente de una o mas entidades de negocio.

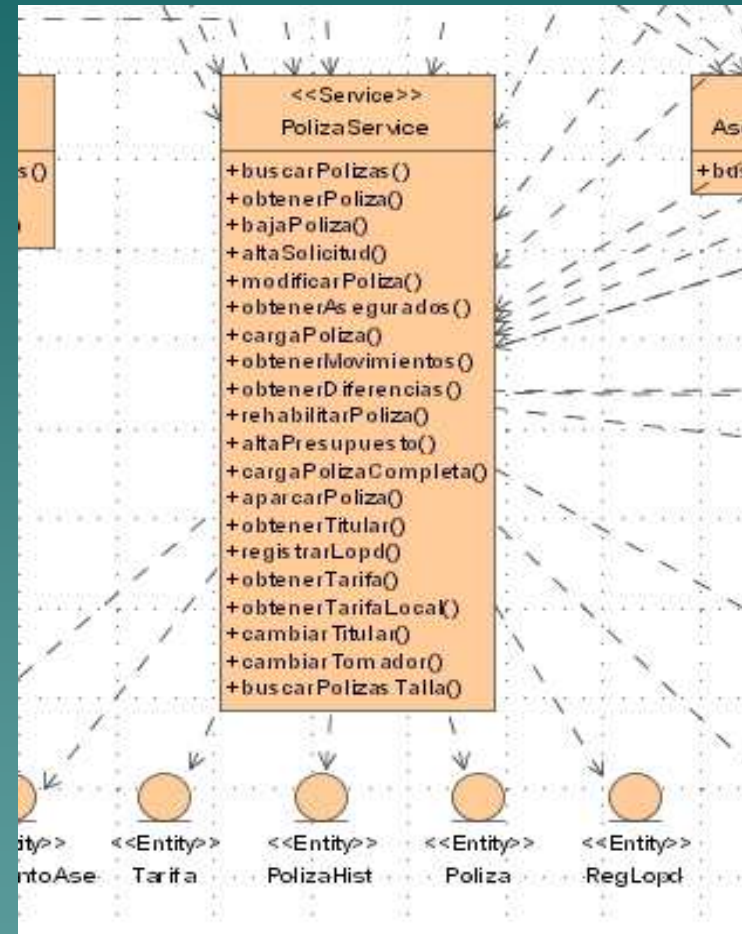
# QUERYS

- Sentencias SQL si conectamos por jdbc.
- Sentencias HSQL si utilizamos Hibernate para persistencia.



# SERVICIOS

- Una estrategia común para separar la capa de presentación del nivel de persistencia, es crear una capa de servicios intermedia.

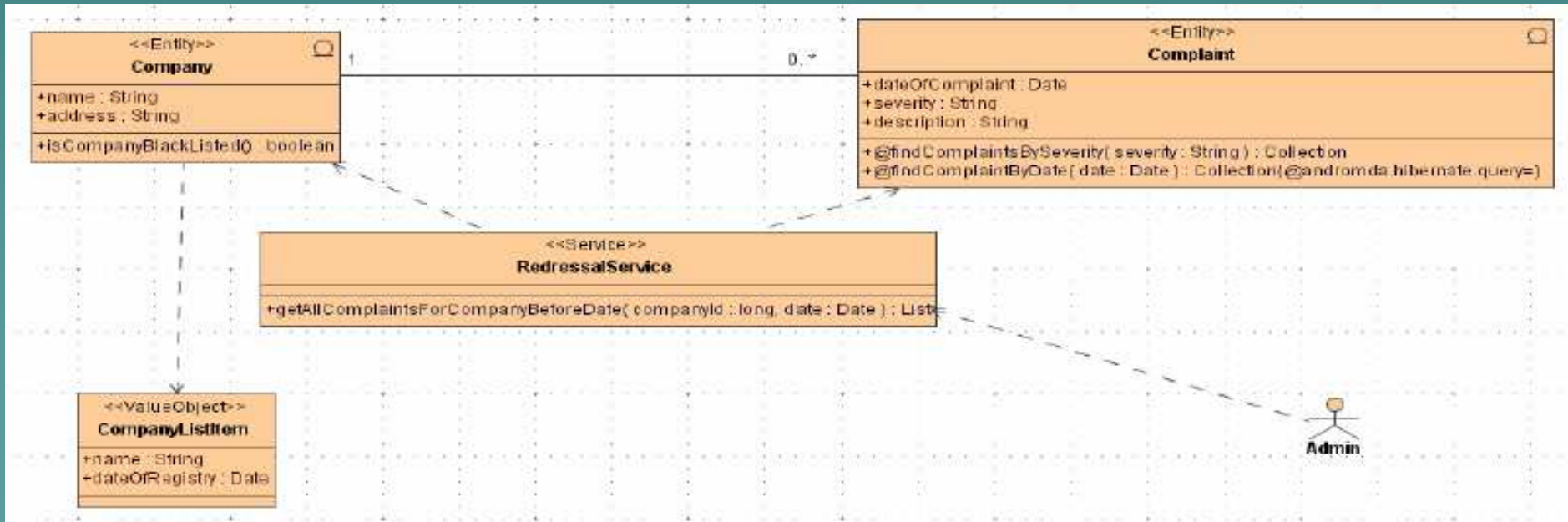


## Generación de código

- En función de cómo hayamos configurado el proyecto de AndroMDA, generará el código sobre una arquitectura u otra.
- AndroMDA nos proporciona el esqueleto sobre el cual implementar las funcionalidades de negocio.
- Este código se implementa en los métodos de las clases Impl generadas por AndroMDA.

# Seguridad

- AndroMDA utiliza ACEGI para implementar la seguridad. Integrable con LDAP.
- El modelo refleja dependencias entre roles y servicios, o entre métodos de un servicio. AndroMDA crea un interceptor que monitoriza todas las llamadas a la capa de servicios.



# AndroMDA y la plataforma AS/400

- Todos los conceptos desarrollados son aplicables.
- Integración con DB2 mediante jdbc, opcionalmente puede utilizarse Hibernate para persistencia.
- DB2 como base de datos, Eclipse y java para desarrollos, WebSphere como servidor web. Integración de herramientas IBM.

<http://www.ibm.com/developerworks/db2/library/techarticle/0306bhogal/0306bhogal.html>

## AndroMDA y la plataforma AS/400

- IBM Toolbox for java & JTOpen
  - Acceso a datos por jdbc
  - Acceso al Integrated File system
  - Llamadas a programas, con parámetros de E/S
  - Llamadas a comandos OS/400 no interactivos
  - Colas de datos
  - Recursos de impresión, spools, colas de salida
  - Información de Jobs
  - Sistema de mensajes
  - Usuarios y grupos
  - DataAreas
  - Valores y estado del sistema

<http://www-03.ibm.com/systems/i/software/toolbox/>