



# MDA: ¿mito o realidad?

Antonio Vallecillo  
Universidad de Málaga

II Jornadas de Ingeniería del Software  
Tenerife, Diciembre 2005  
“Nuevas estrategias para el desarrollo de un software eficiente”



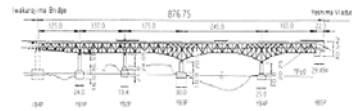
## Agenda

- MDD: Desarrollo de software dirigido por modelos
- MDA: Model Driven Architecture
- MDA: Ventajas, problemas, mitos y realidades
- ¿Conclusiones?

## Los modelos en las ingenierías tradicionales



- Tan antiguos como las Ingenierías (p.e. Vitruvius)
- Los ingenieros “tradicionales” siempre construyen modelos antes de construir sus obras y artefactos
- Los modelos sirven para:
  - **Especificar** el sistema
    - Estructura, comportamiento,...
    - Comunicarse con los distintos *stakeholders*
  - **Comprender** el sistema (si ya existe)
  - **Razonar y validar** el sistema
    - Detectar errores y omisiones en el diseño
    - Prototipado (*ejecutar* el modelo)
    - Inferir y demostrar propiedades
  - **Guiar** la implementación



## Características de los modelos



[Selic, 2003]

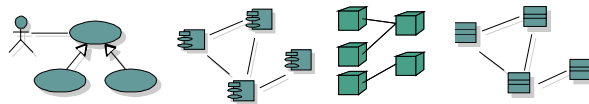
- **Abstractos**
  - Enfatizan ciertos aspectos, mientras que ocultan otros
- **Comprensibles**
  - Expresados en un lenguaje comprensible por los usuarios y *stakeholders*
- **Precisos**
  - Fieles representaciones del objeto o sistema modelado
- **Predictivos**
  - Deben de poder ser usados para inferir conclusiones correctas
- **Baratos**
  - Mas fáciles y baratos de construir y estudiar que el propio sistema



## ¿Qué es un modelo (de software)?



- A *description* of (part of) a system written in a *well-defined language*. (Equivalent to *specification*.) [Kleppe, 2003]
- A *representation* of a part of the *function, structure and/or behavior* of a system [MDA, 2001]
- A *description or specification* of the system and its *environment* for some certain *purpose*. A model is often presented as a combination of drawings and text. [MDA Guide, 2003]
- A set of *statements* about the system. [Seidewitz, 2003]  
(*Statement*: expression about the system that can be considered true or false.)



## Limitaciones actuales de los modelos (de software)



- **Sólo se usan como documentación**
  - Que además no se actualiza!
- **“Gap” entre el modelo y la implementación del sistema**
  - Grandes diferencias semánticas en los lenguajes respectivos
  - No hay herramientas de propagación automática de cambios
    - Cambios en el modelo no se reflejan en el código
    - Cambios en el código no se reflejan en el modelo (el modelo no vuelve a usarse jamás tras la primera implementación)
- **Los distintos modelos del sistema no se armonizan**
  - Suponen vistas de un mismo sistema, pero no hay forma de relacionarlas
  - No hay herramientas de integración de modelos
  - Cada lenguaje de vista tiene una semántica distinta del resto (\*)
- **No hay ni lenguajes ni herramientas para manejar modelos**
  - Solo editores, pero no hay “compiladores”, “optimizadores”, “validadores”, “transformadores de modelos”, etc.
- **¿Estamos realmente hablando de Ingeniería (del software)??**

## The Remarkable Thing about Software



Software has the rare property that it allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods

[John Hogg, 2003]

- Esto facilita enormemente garantizar la fiabilidad entre los modelos y los sistemas producidos, puesto que todos viven en el mismo mundo
- **Corolario:** El modelo **es** la implementación.
- **Salvedad:** Sólo si el modelo contiene toda la información necesaria para producir el sistema

## Model Driven Development (MDD)

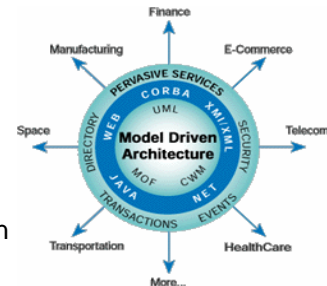


- Un enfoque de desarrollo de software en donde las entidades de primer nivel son los **modelos** y las **transformaciones de modelos**
  - frente a los programas y los compiladores, que constituyeron el paradigma análogo hace treinta años
- MDD implica la generación (casi)**automática** de **implementaciones** a partir de modelos
- En MDD son claves los **lenguajes**, tanto de modelado como de transformación de modelos. Los modelos son conformes a **meta-modelos**
- **MDA** es la propuesta para MDD que hace OMG, usando sus estándares:
  - MOF, UML, OCL, XMI, QVT
  - MOF y UML permiten definir nuevas familias de lenguajes

## Model Driven Architecture



- **MDA es una iniciativa de la OMG**
  - Anunciada en el 2000
  - 10 años de plazo para madurar
  - Debe durar al menos 20 años
- **Extiende OMA**
  - Las plataformas middleware pasan a un segundo plano
  - La clave son los modelos
- **MDA aboga por la separación de la especificación de la funcionalidad de un sistema, independiente de su implementación en cualquier plataforma tecnológica concreta**
- <http://www.omg.org/mda>



## Más allá de la tecnología...



- **Ahora mismo hay demasiadas plataformas y tecnologías**
  - Objetos distribuidos, componentes, servicios web...
  - Realmente no son interoperables entre sí
  - ¿Cuál es la mejor tecnología que debo usar (hoy)?
- **Ademas, la evolución es demasiado rápida**
  - Las tecnologías evolucionan...y se quedan obsoletas muy pronto
  - Conozco la tecnología de hoy pero, ¿cuál va a salir mañana?
  - ¿Y cuánto durará?
  - ¿Como protejo mi inversión en IT frente a estos cambios?
- **Me gustaría separar mis modelos y lógica de negocio de la tecnología concreta en la que están (hoy) implementados**
  - De forma que puedan evolucionar de forma independiente....
    - .... sin tener que tirarlo todo y comenzar de nuevo cada vez
    - .... y de forma que pueda proteger mi inversion en cada uno por separado

## Ventajas (esperadas) de MDA



- **Protege la inversión** ante los continuos cambios en las tecnologías
  - Conserva los PIM de una empresa (su modelo de negocio) cuando aparece nuevo middleware
- **Permite abordar mejor sistemas más complejos**
  - Mediante la separación de diferentes aspectos en diferentes modelos
- **Permite la simulación y la implementación automática de los modelos**
- **Permite la integración de sistemas existentes (COTS, legacy systems)**
  - ADM: Architecture Driven **Modernization**
- **Permite la especificación de los requisitos del sistema independientemente de las plataformas de implementación**
  - MBA: Model-Based **Acquisition**

## Conceptos básicos



- **Computational Independent Model (CIM)**

"A model of the system and its environment, that describes the system requirements but hides the details of its structure and internals"
- **Platform Independent Model (PIM)**

"A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it."
- **Platform Specific Model (PSM)**

"A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform."
- **Platform**

"A set of subsystems/technologies that provide functionality through interfaces and specified usage patterns. It is specified so that any subsystem can use it without being aware of how the functionality provided by the platform is implemented."



## Ejemplos de modelos MDA

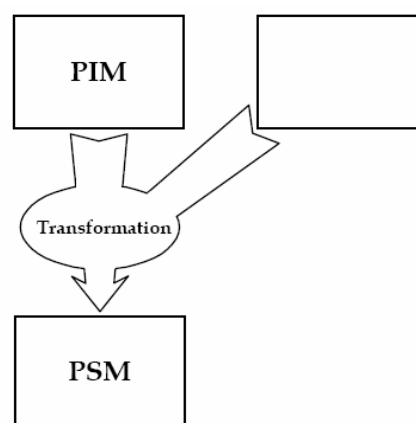


- **CIM**
  - Modelos de casos de uso que capturan los **requisitos** del sistema
- **PIM**
  - La descripción de la **arquitectura software** del sistema, que especifica cómo se descompone la funcionalidad básica del sistema en términos de componentes (arquitectónicos) y conectores
- **PSM**
  - Un **modelo de la implementación J2EE** del sistema, expresado usando el perfil UML para EJB para representar cómo los componentes (arquitectónicos) del sistema se han de implementar como EJBs
- **Código**
  - Los propios componentes EJBs, sus ficheros de configuración, y toda la información necesaria para realizar el deployment en las máquinas concretas

## Transformaciones de Modelos



- Una **transformación de modelos** especifica el proceso de conversión de un modelo a otro (del mismo sistema)
- El **patron MDA** incluye (al menos):
  - un PIM,
  - un modelo de Plataforma,
  - una transformación, y
  - un PSM.



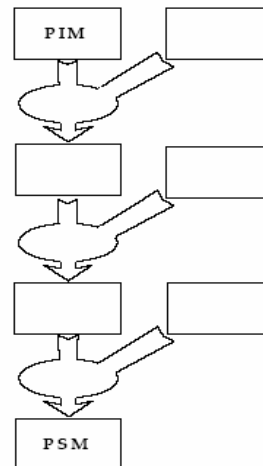


## Aplicaciones sucesivas



- El patron MDA es normalmente utilizado sucesivas veces para producir una sucesión de cambios:

- El PSM resultante de una transformación se convierte en el PIM de la siguiente
- De esta forma, cada “plataforma” se concentra en un aspecto diferente del sistema, y se aplica ordenadamente
- Este proceso es modular y ordenado



## Cómo se construye una aplicación usando MDA



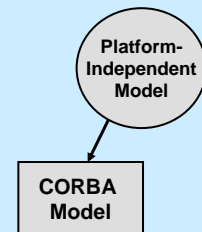
Platform-Independent Model

Un modelo detallado, que especificaría la estructura del sistema, las pre- y post-condiciones en OCL, y el comportamiento en Action Semantics Language (por ejemplo)

Se comienza con el *Platform-Independent Model* (PIM) que representa la lógica del negocio y su funcionalidad, independiente de los detalles de la implementación



## Se genera el PSM



Se escoge una plataforma concreta, y el PIM se transforma al modelo PSM correspondiente a esa plataforma

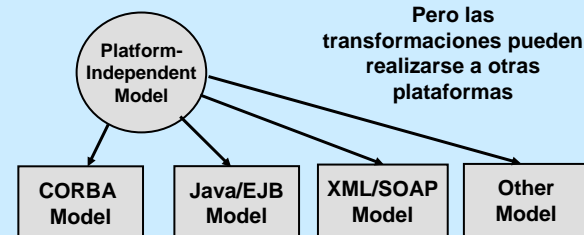
Las transformaciones pueden ser definidas con QVT, entre los metamodelos origen y destino.

Las transformaciones pueden ser parcial o completamente automatizadas

II Jornada Ing. Software, 2017-2018

17

## Generación a múltiples tecnologías



Pero las transformaciones pueden realizarse a otras plataformas

Las transformaciones pueden ser definidas con QVT, entre los metamodelos origen y destino.

Las transformaciones pueden ser parcial o completamente automatizadas

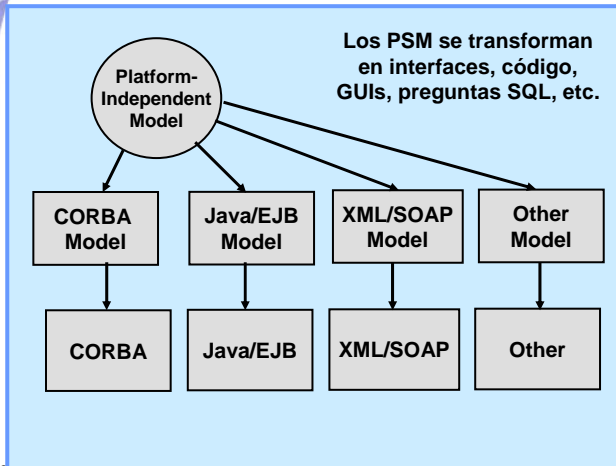
II Jornada Ing. Software, 2017-2018

18

## Generación de implementaciones



~~Write Once, Run Everywhere~~  
Model Once, Generate Everywhere!

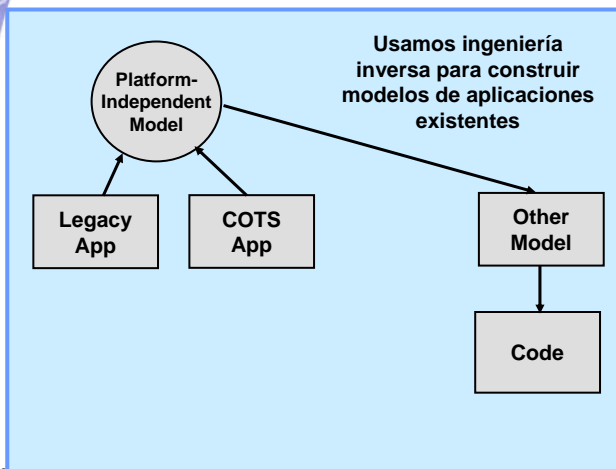


Es fácil contar con implementadores automáticos a partir de modelos específicos, pues son de muy bajo nivel

II Jornada Ing. de Software, 2017-2018

19

## ADM e integración de sistemas



Muy útil para:

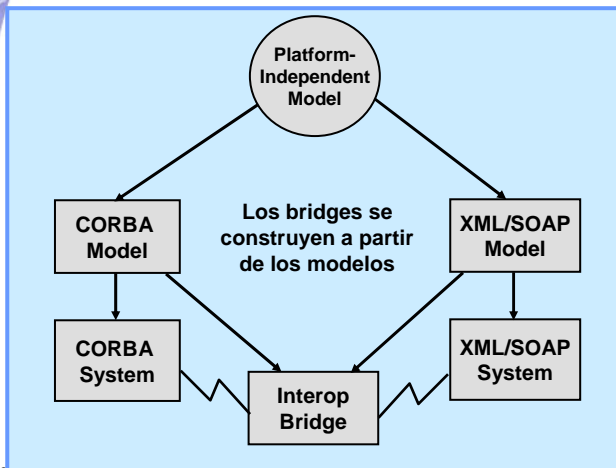
- (1) **Integración** en nuestra aplicación de COTS, sistemas de terceras casas, y sistemas heredados
- (2) **Architecture Driven Modernization:** modernización de sistemas actuales

NASA, DoD, EDF, Banca

II Jornada Ing. de Software, 2017-2018

20

## Generación de bridges



Los bridges (puentes) pueden generarse de forma automática en la mayoría de los casos, tanto dentro de la propia empresa, como para lograr interoperabilidad entre sistemas de diferentes compañías

II Jornadas Ing. Software, Dic. 2005

21

## Ventajas



- **Cada modelo es independiente del resto**
  - Se definen de forma separada
  - Cada modelo define sus propias “entidades”, reside en un nivel de abstracción adecuado, y se expresa en un lenguaje apropiado para el tipo de *stakeholders* interesados en ese tipo de modelo
- **El proceso de desarrollo software se convierte en transformación de modelos**
  - Cada paso selecciona una “plataforma” y transforma uno o mas PIM del sistema en uno (o más) PSM del mismo
  - ...hasta que se llegue a la implementación final del sistema
  - Las transformaciones pueden automatizarse
- **Ganamos modularidad, flexibilidad y facilidad de evolución**
- **Los modelos de la aplicación que capturan la lógica del negocio y la propiedad intelectual se convierten en los principales activos de la empresa, y son independientes de la(s) tecnología(s) en las que serán implementados**

II Jornadas Ing. Software, Dic. 2005

22

## Algunas preguntas interesantes



[Dan Haywood, 2004; Brainbridge, 2005]

1. **¿Sería capaz de adaptarme a esta nueva forma de desarrollo?**
  - ¿Qué haces con tu actual equipo de desarrollo?
  - ¿Estas dispuesto a cambiar tus procesos y herramientas de desarrollo?
  - Además, hacen falta nuevos conocimientos, habilidades, y herramientas!
2. **¿Qué hago con mis programadores?**
  - ¿Tengo que formarlos a todos en esto del modelado?
  - ¿Hace falta despedir a la mitad?
3. **¿Estarían tus accionistas dispuestos a un cambio tan radical?**
  - ¿Cuánto me cuesta? ¿Se justifica el gasto?
4. **¿Realmente cambio tan a menudo de plataforma y tecnologías?**
5. **¿Tendremos que tirar todo nuestro código actual?**
6. **¿El código que se genera es eficiente?**
7. **¿El código que se genera es legible y mantenible?**
8. **¿Es MDA aplicable a todos los dominios de aplicación?**
9. **¿Tenemos disponibles herramientas MDA ahora mismo?**

II Jornadas Ing. Software, Dic. 2005

23

## Algunas preguntas interesantes



[Dan Haywood, 2004; Brainbridge, 2005]

10. **Modelar la estructura del sistema es fácil, pero....**
  - ¿Qué pasa con el comportamiento?
    - State-based approaches (state machines, ASL, SDL)
    - Declarative approaches (pre-post, contracts)
    - Protocol-based approaches (sequence and interaction diagrams)
  - Que pasa con otros aspectos muy importantes como los estilos arquitectónicos, Calidad de Servicio, GUIs,...
11. **¿Y los lenguajes de modelado?**
  - MOF (OMG) vs. EMF (Microsoft), o UML profiles vs. DSLs.
  - ¿Tienen los lenguajes y estándares de OMG suficiente poder expresivo para cualquier dominio de aplicación?
12. **¿De verdad funciona esto del MDD?**
  - ¿Sinceramente, tu crees en eso de pintar dos cajas y tres líneas y obtener todo el código de tu aplicación?
  - ¿Seremos capaces de generar código a partir de modelos, incluso cuando todavía no estamos de acuerdo ni en cómo representar el comportamiento?
  - ¿No te preocupa un poco que haya dos escuelas distintas?
    - Los **translacionistas** (e.g. Executable UML): PIMs 100% ejecutables
    - Los **elaboracionistas** (e.g. Optimal/J, ArcStyler): PIMs 70% ejecutables, completados manualmente.

II Jornadas Ing. Software, Dic. 2005

24

## Algunas respuestas (I)



### 1. ¿Sería capaz de adaptarme a esta nueva forma de desarrollo?

- Realmente, trabajar con modelos es más fácil que trabajar con código:
  - mayor nivel de abstracción,
  - herramientas visuales,
  - lenguajes de dominio específico (UML profiles o EMF's DSL).
- Las metodologías de desarrollo no son tan diferentes
- Los procesos de desarrollo son análogos

### 2. ¿Qué hago con mis programadores?

- Muchos programadores disfrutaban más modelando que programando
- También se necesitan muy buenos programadores para configurar y “afinar” las transformaciones de modelos de forma que sean muy eficientes, y también para diseñar y depurar patrones eficientes de traducción



### ¿No suenan “familiares” tanto las preguntas como las respuestas?

**Hace unos años hicimos estas mismas preguntas al pasar de lenguajes de bajo nivel a la programación estructurada, y luego las volvimos a hacer antes de adoptar la programación orientada a objetos...**

## Algunas respuestas (II)



### 3. ¿Estarían tus accionistas dispuesto a un cambio tan radical?

- Según todos los informes y experiencias, la productividad aumenta entre un 200% y un 1000%, lo que va a permitir asumir los costes añadidos del modelado inicial
- También voy a ahorrar otros costes, como los asociados a la documentación del código (el modelo es la documentación)
- Los costes en formación los tengo que pagar de todas formas, y se equilibran con el tiempo:
  - O formo a mis programadores actuales en los nuevos sistemas, o formo al personal que voy contratando en la tecnología obsoleta que uso.

### 4. ¿Realmente cambio tan a menudo de plataforma y tecnología?

- No cada 3 o 4 años, pero MDA esta pensada para soportar iteraciones cada 20 o 30 años, mientras que la vida útil de las tecnologías convencionales es de unos 10 años.

### 5. ¿Tendremos que tirar todo nuestro código actual?

- Ni mucho menos. Tal y como se realizan en todas las experiencias de MDA, normalmente se modelan los nuevos módulos y aplicaciones, y se van construyendo “bridges” con los componentes de las aplicaciones existentes. Poco a poco esas aplicaciones pueden ser modeladas y su nuevo código generado a partir de los modelos.

### ¿No suenan “familiares” las preguntas y las respuestas? :-)

## Algunas respuestas (III)



6. **¿El código que se genera es eficiente?**
  - La eficiencia depende del modelo y de las reglas de transformación que se definan.
  - En teoría, aparecerán también optimizadores...
7. **¿El código que se genera es legible y mantenible?**
  - Los generadores pueden configurarse para insertar comentarios en el código
  - De todas formas, ¿se preocupa Vd. de la legibilidad del ensamblador que produce su compilador?
8. **¿Es MDA aplicable a todos los dominios de aplicación?**
  - En teoría sí, pero en la práctica depende de los lenguajes de modelado disponibles para su dominio.
9. **¿Tenemos disponibles herramientas MDA ahora mismo?**
  - Sí (<http://www.omg.org/mda>). Aunque desde mi punto de vista (personal) están en un estado un tanto inmaduro todavía.

**¿No suenan “familiares” las preguntas y las respuestas? :-)**

## Algunas respuestas (IV)



10. **Modelar la estructura del sistema es fácil, pero ¿qué pasa con el comportamiento y otros aspectos no funcionales?**
  - **Touché!**
  - Respecto al comportamiento, es posible resolver el problema de forma satisfactoria si fijamos el dominio de aplicación
  - Los otros aspectos tienen ahora mismo soluciones ad-hoc
11. **¿Y los lenguajes de modelado?**
  - Pueden usarse lenguajes específicos para cada dominio de aplicación
  - Pero es **MUY** importante considerar la interoperabilidad entre ellos (no sólo la semántica, sino entre las herramientas que manejan esos modelos: editores, transformadores, etc.)
  - **En ese sentido, ojo a la batalla OMG (MOF) vs. Microsoft (EMF)**

## Y la pregunta final...



### 12. ¿De verdad funciona esto del MDD?

- ¿Sinceramente, tu crees en eso de pintar dos cajas y tres líneas y obtener todo el código de tu aplicación?
- ¿Seremos capaces de generar código a partir de modelos, incluso cuando todavía no estamos de acuerdo ni en cómo representar el comportamiento?

- **La mejor respuesta a esta pregunta la dan las experiencias que actualmente han demostrado que (en ciertos dominios de aplicación) MDA no sólo es factible, sino que consigue mejoras espectaculares en la productividad y la calidad de los productos desarrollados**



## Éxitos (hasta la fecha)



- Lockheed Martin (F16 mission computer)
- Nortel (Passport)
- TRW Automotive
- BAE Systems (Stingray torpedo)
- US DoD SIAP (Single Integrated Air Picture)
- US Government Model-based Acquisition
- US ERA (Electronic Record Archives)

### ● Usando herramientas y sistemas de:

- Kennedy Carter iUML
  - MDA y ADM usando UML ejecutable
- IO-Software ArcStyler, Compuware Optimal/J, Borland Together, etc.
  - Herramientas MDA de carácter general, muy útiles para aplicaciones distribuidas (J2EE o CORBA), aplicaciones Web, o aplicaciones orientadas a servicios.
- Kabira's Adaptive Realtime Infrastructure (ARI)
  - Aplicaciones para sistemas distribuidos transaccionales
- Secant technologies
  - Aplicaciones de extracción de conocimiento





## Éxitos (MDD y MDA en Canarias)



- MDD y MDA llevan aplicándose con éxito en Canarias desde hace algunos años
- **Ejemplo: Open Canarias, S.L.**
  - Consejería de Política Territorial
    - Proyecto **CENTINELA** (J2EE)
    - Proyecto **MAYTE** (Medio Ambiente y Territorio Electrónico) (J2EE)
  - Servicio Canario de Salud
    - **Registro y listas de espera renal y trasplantes**
  - Cabildo de Tenerife
    - **Contratos de Obras para Deportes** (J2EE)
  - Diputación de Guipúzcoa
    - **Integración de CICS DB2 con aplicaciones web** (J2EE)

## Mitos falsos sobre MDA



- **MDA es un “silver bullet” que sirve para desarrollar cualquier tipo de aplicación**
  - **FALSO.** Es muy aplicable en ciertos dominios de aplicación. En otros, por supuesto, no lo es tanto
- **MDA ya está terminado y todos sus componentes listos**
  - **FALSO.** Ahora sólo contamos con lo más básico. Todavía faltan por llegar bastantes piezas y otras tienen que madurar bastante (QVT).
- **MDA nunca sucederá de verdad**
  - **FALSO.** Ya está sucediendo de verdad
- **MDA requiere un desarrollo en “cascada”**
  - **FALSO.** MDA es independiente de la metodología de desarrollo. De hecho, se está usando con éxito con metodologías “ágiles” (XP)
- **MDA necesitas herramientas muy caras**
  - **FALSO.** Hay incluso herramientas muy interesantes que son open source
- **MDA generará absolutamente todo a partir del PIM**
  - **FALSO.** Sólo en algunos dominios específicos eso es cierto. En el resto será necesario “inyectar” muchas partes del código, configurar la aplicación,... y otras muchas tareas de forma manual

## ¿Conclusiones?



- **MDD eleva el nivel de abstracción de programas a modelos**
  - Igual que los lenguajes de programación estructurada elevaron el nivel de abstracción del ensamblador
- **MDA es la propuesta de OMG para hacer MDD, usando sus estándares: UML, MOF, XMI, OCL, QVT**
- **Las ideas son buenas, los primeros resultados están aquí (¡y son buenos!) y la dirección parece la adecuada**
- **Algunos peros:**
  - las herramientas y la tecnología que soporta MDA no están del todo maduras
  - La compatibilidad y portabilidad entre modelos no funciona del todo
  - Hay pocas experiencias todavía
- **De todas formas, ¿se plantearía a día de hoy si usar o no lenguajes de programación, frente a ensambladores, para construir sus aplicaciones?**
- **¿Tendré otra opción (que no pase por el modelado) cuando la complejidad de las aplicaciones siga creciendo?**
- **¿Merece la pena dar el salto? ¿hacia dónde?..... 😊**

II Jornadas Ing. Software, Dic. 2005

33

## ¡Gracias!



av@lcc.uma.es  
<http://www.lcc.uma.es/~av>