Universidad Católica Nuestra Señora de la Asunción
Facultad de Ciencias y Tecnología
Ingeniería Informática

Trabajo práctico
De
Teoría y Aplicación de la Informática 2

Tema:

MDA
Model Driven Architecture
Arquitectura Dirigida por Modelos

MDD-
Model Driven Development
Desarrollo Dirigido por Modelos

Raúl Orué Rotela

Año 2007

Introducción a MDA

Evolución del desarrollo de aplicaciones

- **1960's** Lenguajes ensambladores / máquina (dep. del **procesador**). 1ros. Lenguajes de alto nivel: 2GL: Fortran, Cobol. 3GL: Algol, PL/1, Basic

- **1970's**Dominio de Lenguajes imperativos (Pascal, **C**) (3GL). Procesamiento por lotes. Aparece el primer lenguaje OO: Smalltalk.

- **1980's**Lenguajes 3GL. Se difunden los lenguajes OO. Nace **C++**

- **1990's**Proliferación de *middlewares* (CORBA, IDL, DCOM, etc.) y APIs. Surge **Java**. Lenguajes de marcado (**HTML**, **XML**) y de scrips (JavaScrip, Perl). Leng. orientados a la web (**PHP**). Leng. de modelado (**UML**) y consolidación de los **lenguajes OO**.

- **2000's**Nuevos y estandarizados protocolos (ej. SOAP p/ *webservices*) y plataformas (**J2EE**, **.NET**). Herramientas **MDD** generadoras de código. Estandarización del MDD: ***Model Driven Architecture***. Modelado de sistemas se convierte en algo normal y habitual.

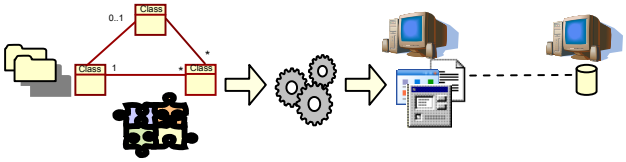
- **2010's**¿Cuál será la *próxima 'gran cosa'*?

A qué uno se enfrenta cuando desarrolla SW hoy?

→ La heterogeneidad es permanente.
→ Debe existir consenso en las interfaces y la interoperabilidad.
→ Debería preservarse la inversión en el desarrollo de software.
→ Necesidad de representar la lógica de una aplicación independiente de una tecnología particular.
→ Reusabilidad de componentes SW, código y *expertise* del personal IT.
→ Contemplar robustez, escalabilidad y seguridad al principio del diseño de una solución SW (aplicación), que permita incorporar rápidamente nuevos cambios en el futuro y pueda ser adaptable a los cambios tecnológicos de plataformas.

¿A qué uno se enfrenta cuando desarrolla SW hoy?

Model Driven Development Approaches…



Estandarizar el desarrollo dirigido por modelos (MDD) definiendo una arquitectura que utilice lenguajes estándares para la creación de modelos y su transformación en modelos y código de una plataforma específica.

La solución a esos problemas planteado desde inicios de los años '90:

- *Desarrollar una arquitectura basada en la tecnología de objetos para la integración de aplicaciones que garanticen:*
  - *reusablidad de componentes;*
  - *interoperabilidad y portabilidad;*
  - *basada en software disponible comercialmente.*
- *Que las especificaciones estén disponible gratuitamente.*
- *Que existan implementaciones (herramientas que soporten la arquitectura).*
- *Que sea un estándar de producción donde no exista una compañía controladora (propietaria) ni unas pocas sean las beneficiadas*.

Estandarizar el desarrollo dirigido por modelos (MDD) definiendo una arquitectura que utilice lenguajes estándares para la creación de modelos y su transformación en modelos y código de una plataforma específica.

La arquitectura dirigida por modelos (Model Driven Arquitectura) es una especificación detallada por el OMG (Object Management Group) que integra diferentes especificaciones y estándares definidos por la misma organización con la finalidad de

ofrecer una solución a los problemas relacionados con los cambios en los modelos de negocio, la tecnología y la adaptación de los sistemas de información a los mismos.

MDA nos permite el despliegue de aplicaciones empresariales, diseñadas sin dependencias de plataforma de despliegue y expresado su diseño mediante el uso de UML y otros estándares, potencialmente en cualquier plataforma existente, abierta o propietaria, como servicios web, .Net, Corba, J2EE, u otras. La especificación de las aplicaciones y la funcionalidad de las mismas se expresa en un modelo independiente de la plataforma que permite una abstracción de las características técnicas específicas de las plataformas de despliegue.

Mediante transformaciones y trazas aplicadas sobre el modelo independiente de la plataforma se consigue la generación automática de código específico para la plataforma de despliegue elegida, lo que proporciona finalmente una independencia entre la capa de negocio, y la tecnología empleada. De esta manera es mucho más simple la incorporación de nuevas funcionalidades, o cambios en los procedimientos de negocio sin tener que llevar a cabo los cambios en todos los niveles del proyecto.

Simplemente se desarrollan los cambios en el modelo independiente de la plataforma, y éstos se propagarán a la aplicación, consiguiendo por tanto una considerable reducción del esfuerzo en el equipo de desarrollo, en los errores que tienden a producirse en los cambios introducidos en las aplicaciones mediante otros métodos de desarrollo, y por consiguiente, la reducción de costes y aumento de productividad que conlleva, tan demandados tanto la industria de desarrollo de software como el resto de las empresas.

MDA se apoya sobre los siguientes estándares para llevar a cabo su función: - UML: empleado para la definición de los modelos independientes de la plataforma y los modelos específicos de las plataformas de destino. Es un estándar para el modelado introducido por el OMG. - MOF: establece un marco común de trabajo para las especificaciones del OMG, a la vez que provee de un repositorio de modelos y metamodelos. - XMI: define una traza que permite transformar modelos UML en XML para poder ser tratados automáticamente por otras aplicaciones. - CWM: define la transformación de los modelos de datos en el modelo de negocio a los esquemas de base de datos.

Conceptos de MDA

De cara a entender MDA y sus características, su funcionamiento y su aplicación al proceso de desarrollo, revisaremos los conceptos básicos de MDA y su forma de uso.

Sistema

Los conceptos de MDA se definen centrados en la existencia o planteamiento de un sistema, que puede contener un simple sistema informático, o combinaciones de componentes en diferentes sistemas informáticos, o diferentes sistemas en diferentes organizaciones, etc

Modelo

Un modelo de un sistema es una descripción o una especificación de ese sistema y su entorno para desempeñar un determinado objetivo. Los modelos se presentan

normalmente como una combinación de texto y dibujos. El texto se puede presentar en lenguaje de modelado, o en lenguaje natural.

Dirigido por modelos
MDA es un acercamiento al desarrollo de sistemas, que potencia el uso de modelos en el desarrollo. Se dice que MDA es dirigido por modelos porque usa los modelos para dirigir el ámbito del desarrollo, el diseño, la construcción, el despliegue, la operación, el mantenimiento y la modificación de los sistemas.

Arquitectura
La arquitectura de un sistema es la especificación de las partes del mismo, las conexiones entre ellos, y las normas de interacción entre las partes del sistema haciendo uso de las conexiones especificadas. MDA determina los tipos de modelos que deben ser usados, como preparar dichos modelos y las relaciones que existen entre los diferentes modelos.

Punto de vista
Un punto de vista es una abstracción que hace uso de un conjunto de conceptos de arquitectura y reglas estructurales para centrarse en aspectos particulares del sistema, obteniendo un modelo simplificado.

Vista
Una vista es una representación del sistema desde un determinado punto de vista.

Plataforma
Una plataforma es un conjunto de subsistemas y tecnologías que aportan un conjunto coherente de funcionalidades a través de interfaces y determinados patrones de uso, que cualquier aplicación que se construya para esa plataforma puede usar sin preocuparse por los detalles de la implementación o como se lleva a cabo la misma dentro de la plataforma.

Aplicación
En MDA se define el término aplicación como una funcionalidad que tiene que ser desarrollada. Por tanto podemos definir un sistema en términos de la implementación de una o más aplicaciones, soportadas por una o más plataformas.

Independencia de la plataforma
La independencia de la plataforma es una cualidad que tienen que presentar los modelos. Lo que significa que un modelo es independiente de las facilidades o características que implementan las plataformas, de cualquier tipo

Puntos de vista MDA
MDA establece tres puntos de vista que se emplearán a lo largo del proceso de ingeniería:

- Punto de vista independiente de la computación: se centra en el entorno del sistema y los requisitos para el mismo. Los detalles de la estructura y procesamiento del sistema no se muestran, o aún no están especificados.
- Punto de vista independiente de la plataforma: se centra en las operaciones del sistema, mientras oculta los detalles necesarios para una determinada plataforma. Muestra aquellas partes de la especificación del sistema que no

cambian de una plataforma a otra. En este punto de vista debe emplearse lenguaje de modelado de propósito general, o bien algún lenguaje específico del área en que se empleará el sistema, pero en ningún caso se emplearán lenguajes específicos de plataformas.
- Punto de vista de plataforma específica: combina el punto de vista independiente de la plataforma con un enfoque específico para su uso en una plataforma específica en un sistema.

Modelo independiente de la computación
Un modelo independiente de la computación (CIM) es una vista del un sistema desde el punto de vista independiente de la computación. En algunos casos, nos referimos al modelo independiente de la computación como el modelo del dominio, y se usa vocabulario propio de los expertos en el dominio para la especificación

Modelo independiente de la plataforma
Un modelo independiente de la plataforma (PIM) es una vista del sistema desde el punto de vista independiente de la plataforma. Expone un carácter independiente de la plataforma sobre la que se desplegará, de modo que pudiera ser empleado en diferentes plataformas de carácter similar.

Una técnica común para alcanzar el grado de independencia de la plataforma necesario es definir un sistema basado en una máquina virtual que abstraiga los modelos particulares de las plataformas existentes y sea neutral respecto a las mismas.

Modelo específico de la plataforma
Un modelo específico de la plataforma (PSM) es una vista de un sistema desde el punto de vista dependiente de la plataforma. Combina las especificaciones del modelo independiente de la plataforma con los detalles que especifican el uso de una plataforma específica por parte del sistema.

Modelo de plataforma
Un modelo de plataforma expone un conjunto de conceptos técnicos que representan las diferentes formas o partes que conforman un sistema, y los servicios que provee. También expone, para su uso en los modelos específicos de la plataforma, conceptos que explican los diferentes elementos que provee una plataforma para la implementación de una aplicación en un sistema.

Una modelo de plataforma incluye también la especificación de requisitos en la conexión y uso de las partes que integran la plataforma, y la conexión de aplicaciones a la plataforma.

Transformación de modelos
La transformación de modelos es el proceso de convertir un modelo en otro modelo del mismo sistema.

La transformación de un modelo independiente de la plataforma en un modelo dependiente de la plataforma no es necesaria para PIM basados en una máquina virtual.

En este caso hay que transformar el PIM correspondiente a la máquina virtual en un modelo de plataforma específico

Servicios penetrantes

Los servicios penetrantes son servicios que están disponibles un amplio catálogo de plataformas. MDA proveerá servicios penetrantes comunes e independientes de la plataforma y dispondrá de trazas para la transformación de los modelos, que permitirá la transformación de los servicios expuestos en los PIMs a las plataformas de destino.

Usando MDA

Expuestos los conceptos básicos de MDA, es necesario conocer como se relacionan unos modelos, su modo de uso y las transformaciones entre PIM y PSM.

El modelo de origen es el CIM, con el que se modelan los requisitos del sistema, describiendo la situación en que será usado el sistema y que aplicaciones se espera que el sistema lleve a cabo, sirviendo tanto como ayuda para entender el problema como una base de vocabulario para usar en los demás modelos.

Los requisitos recogidos en el CIM han de ser trazables a lo largo de los modelos PIM y PSM que los implementan.

El CIM puede consistir en un par de modelos UML que muestren tanto la distribución de los procesos (ODP, Open Distributed Processing) como la información a tratar. También puede contener algunos modelos UML más que especifiquen en más detalle los anteriores.

A partir del CIM, se construye un PIM, que muestra una descripción del sistema, sin hacer referencia a ningún detalle de la plataforma. Debe presentar especificaciones desde el punto de vista de la empresa, información y ODP.

Un PIM se puede ajustar a un determinado estilo de arquitectura, o a varios. Después de la elaboración del PIM, el arquitecto debe escoger una plataforma (o varias) que satisfagan los requisitos de calidad.

Mapas de transformación

Mediante mapas, MDA especifica las reglas de transformación de un PIM a un PSM para una plataforma en concreto. Estos mapas incluyen la transformación de tipos de valores para la transformación desde el PIM al PSM, los metamodelos y sus reglas para la transformación en tipos de valores existentes en las plataformas.

Cuando se prepara un modelo haciendo uso de un leguaje independiente de la plataforma, especificado en un metamodelo y posteriormente se elige una plataforma para el despliegue, debe existir una especificación de transformación entre el metamodelo independiente de la plataforma y el metamodelo que describe la plataforma

Una forma de facilitar la transformación de modelos es la identificación de elementos en el PIM que deben transformarse de una manera concreta para la plataforma de destino, y marcarlos como tal. Una marca expresa un concepto del PSM en el PIM; las marcas alejan el PIM de la independencia de la plataforma, por lo que un arquitecto debe

mantener un PIM limpio, marcarlo para su adaptación a una plataforma en concreto. Las marcas deben concebirse como una capa transparente que se pone sobre el modelo.

Una vez es elegida la plataforma, existe un mapa para la transformación de modelos. Este mapa incluye un conjunto de marcas, que usamos para marcar los elementos del PIM como guía para la transformación del modelo.

Las marcas pueden definir tipos del modelo, especificación de clases, asociaciones, roles, estereotipos,… las marcas también pueden especificar características cualitativas que después en la transformación se convertirá en el objetivo apropiado para el cumplimiento de los requisitos.

Las marcas deben tener un modelo y una estructura que permita mantener la coherencia, que impida el marcado de un elemento con marcas mutuamente excluyentes.

Los mapas de transformación pueden mantener también plantillas, que son modelos parametrizados que especifican tipos concretos de transformaciones.

Podemos asociar un conjunto de marcas a una plantilla para identificar instancias en un modelo que deben ser transformados de acuerdo a las indicaciones de la plantilla.

Un elemento en un modelo puede ser marcado varias veces, empleando marcas procedentes de diferentes plantillas, y de distintos mapas de transformación, por lo que esos elementos serán transformados tantas veces como marcas tengan, siguiendo las reglas que especifican los mapas y las plantillas para los que fueron marcados.

Existe la posibilidad de incluir información relativa a patrones que extienda las características y tipos de los metamodelos y el lenguaje de la plataforma específica elegida para el despliegue

El uso de información adicional para la transformación implica la necesidad de información de los patrones para la transformación, que serán específicos para la plataforma de destino.

Transformaciones

El siguiente paso al establecimiento de las marcas en los modelos y la selección de mapas de transformación es aplicar la transformación desde el PIM marcado al PSM. Este proceso puede ser manual, asistido por computador, o automático.

La transformación es el proceso que toma como entrada el PIM marcado y da como resultado el PSM del sistema, y el registro de transformación. Algunas herramientas pueden hacer una transformación del PIM directamente a código desplegable en la plataforma de destino o incluso conceptos como MDR (Model-Driven Runtime Environment) que propone el uso de un entorno de ejecución para los PIM, directamente, sin generación de PSM ni código para la plataforma. En cualquier caso el OMG sostiene que la transformación debe emitir un PSM para ayudar al entendimiento del código y la depuración del mismo.

El registro de transformación incluye una traza desde cada elemento del PIM a los elementos correspondientes en el PSM, especificando que parte de los mapas de transformación fueron empleados para derivar los elementos del PSM desde los elementos del PIM. Una herramienta que mantenga los registros de transformación debe ser capaz de sincronizar de forma automática los cambios producidos en un modelo al otro.

El PSM producido por una transformación es un modelo del mismo sistema que ha sido especificado en el PPIM. También especifica como el sistema hace uso de una determinada plataforma.

Un PSM será una implementación del sistema si proporciona toda la información necesaria par construir el sistema y ponerlo en marcha.

La transformación de modelos convierte un modelo de un sistema en otro modelo del mismo sistema pero que tiene mayor o menor nivel de abstracción. Por ejemplo, en los modelos de MDA se transforma un modelo independiente de la plataforma en uno específico de la plataforma añadiendo información que permita trazar ambos modelos. La transformación de PIM a PSM puede llevarse a cabo de diferentes maneras: Construyendo manualmente el PSM a partir del PIM

De forma semiautomática, generando el PSM esqueleto que es completado a mano. De forma totalmente automática, generando un PSM completo a partir del PIM. Para este último método se utilizan herramientas especializadas que implementan diferentes algoritmos de transformación para pasar de un tipo de modelo a otro, y forman parte de uno de los pilares de MDA. Estas herramientas son las que evaluaremos mas adelante en nuestro trabajo. Aunque hayamos identificado las vistas principales de MDA, es difícil identificar cuando un modelo pasa de ser por ejemplo PIM a ser PSM ya que las transformaciones se hacen de manera gradual, refinando los modelos haciendo transformaciones sucesivas hacia modelos de mas bajo nivel hasta obtener un modelo que puede ser transformado en el modelo siguiente de manera directa.. Es decir, que para llegar al “PSM” se necesitarán antes varias transformaciones PIM→PIM. Gracias a estas herramientas de transformación se obtiene gran parte del código que implementa el sistema para la plataforma elegida, partiendo de un PSM. El desarrollador solamente deberá añadir la funcionalidad que no pueda representar mediante el PIM o el PSM. Podemos reconocer entonces dos tipos distintos de transformación:

Transformaciones de Tipos *(Model Type Mapping):* segunda especificación, “un mapping de tipos especifica un mapping para transformar cualquier modelo construido con tipos del PIM a otro modelo expresado con tipos del PSM”. Es decir que a cada tipo de elemento del PIM se le aplica una regla determinada para transformarlo en uno o más elementos de un PSM. Esto se puede hacer definiendo estereotipos (*Por ejemplo, cada instancia de las clases con el estereotipo Segment en el PIM se van a transformar en un EJB Entity del PSM EJB*) o definiendo reglas en función de valores reinstancias en el PIM (*Por ejemplo: Para cada atributo con valor “public” en la visibilidad, transformarlo en “private” y agregarle los getters y setters correspondientes.*)

Transformaciones de instancia *(Model Instante Mapping):* Son los elementos propios del PIM que deben ser transformados de una manera particular, para una plataforma determinada, esto se consigue mediante el uso de las **marcas**. El desarrollador mantiene un PIM puro, sin marcas y luego genera los PIM's marcados que van a ser específicos de una plataforma dada. *Un ejemplo de esto podría ser para la plataforma EJB la marca Entity que si esta activa en alguna clase del PIM indicará que esa clase se transformará en un EJB Entity.* Como se puede apreciar, las marcas le permiten al desarrollador dirigir o controlar las transformaciones hacia una plataforma determinada.

También se deben generar **puentes de comunicación** entre los distintos modelos, tanto PSM como de PSI para aclarar el funcionamiento de estos puentes vemos el siguiente ejemplo: *Tenemos un PIM original del cual se generaron tres PSM's uno de ellos que especifica y describe las interfaces Web del sistema, otro especifico para la plataforma EJB y el último basado en modelos relacionales que describe el esquema de la base de datos mediante un diagrama de entidad - relación. En este ejemplo los puentes de comunicación se utilizarían entre PSM's y PSI's para permitir a la capa Web comunicarse con los componentes*

EJB y a estos con la base de datos del sistema. Para pasar de un nivel a otro se necesitaran lenguajes para permitir transformaciones entre modelos como por ejemplo QVT, y consistencia, sincronización y trazabilidad entre los modelos.

Para poder cumplir con el enfoque propuesto por MDA, es conveniente que el proceso de transformación disponga de una serie de propiedades o características. En el capitulo 7 de *MDA Explained, de* Kleppe, A; et al. Se exponen las características deseables que deben tener las transformaciones en MDA. Estas propiedades son , en orden de importancia:

Posibilidad de ajustar las transformaciones: permite al desarrollador tener cierto control sobre el proceso de transformación, esto puede lograrse mediante el ***control manual:*** permitiendo al usuario controlar directamente la transformación definiendo los elementos del modelo que se transformarán según reglas determinadas, siendo ésta la manera mas flexible y a su vez mas complicada y propensa a errores, y por ende la menos utilizada; mediante ***condiciones en las transformaciones:*** el desarrollador agrega condiciones a cada regla de transformación determinando en que casos debe aplicarse dicha regla; o mediante ***parámetros de transformación.*** **Trazabilidad:** Implica el conocimiento del elemento del modelo de origen a partir del cual se generó cualquier elemento del modelo destino. Este es un punto importante para la búsqueda y control de errores y para cumplir con las próximas dos características: el control incremental y la bidireccionalidad.

Consistencia incremental: Un proceso de transformación incremental sabe qué elementos del modelo de destino debe reemplazar por los nuevos mientras mantiene la información extra (cambios puntuales y específicos que el desarrollador pudiera oportunamente haber realizado sobre este.) del modelo de destino sin modificaciones.
Bidireccionalidad: Significa que las transformaciones puedan operar en ambas direcciones. Un estándar actualmente utilizado en la definición de transformaciones es el **QVT (Query, Views and Transformations).** El QVT define el modo en que se llevan a

cabo las transformaciones entre modelos cuyos lenguajes han sido definidos usando MOF. Consta de tres partes:

Un lenguaje para crear vistas de un modelo. Propone el uso de consultas para la creación de vistas de un modelo.

Un lenguaje para realizar consultas sobre modelos. El OCL 2.0 ya que los usuarios están familiarizados con él, evita la creación de un nuevo lenguaje y además ya existe soporte para OCL en muchas herramientas.

Un lenguaje para escribir definiciones de transformaciones. Esta última parte es la más relevante para MDA, porque actualmente no existe un estándar que especifique la definición de transformaciones entre modelos. Propone dos tipos de transformaciones:

Relaciones: son especificaciones de transformaciones multidireccionales, sin la capacidad de modificar el modelo, sin embargo pueden chequear la consistencia entre dos o más modelos relacionados, Se utilizan en la especificación del desarrollo de un sistema o para chequear la validez de un mapping.

Mappings: Implementación de transformaciones y a diferencia de las relaciones, los mappings son unidireccionales y pueden devolver valores y refinar una o varas relaciones con las cuales debe ser consistente.

D. Estándares sobre los que se apoya MDA Metamodelos: El metamodelado es un mecanismo que permite definir formalmente lenguajes de modelado. Básicamente se trata de utilizar modelos para describir otros modelos. Por ejemplo, el metamodelado de UML define los conceptos y reglas que se necesitan para crear modelos UML.

Existen cuatro capas de modelado definidas por el OMG: ***M0 – instancias***, son las instancias reales del sistema; ***M1 – Modelos***, incluye las entidades del modelo del sistema; ***M2 – Metamodelo***, incluye las entidades de un lenguaje de modelado y por último el ***M3 – Meta-metamodelo***, que contiene las entidades que definen los lenguajes de modelado.

El metamodelado es importante para MDA ya que permite definir lenguajes de modelado no ambiguos de manera tal que las herramientas puedan leer escribir y comprenderlos. También utiliza el metamodelado para definir las transformaciones. Uno de los estándares más importantes utilizados en MDA es el **MOF (Meta Object Facility),** fue definido por el OMG y pertenece a la capa M3**.** Utiliza 5 construcciones básicas para definir un lenguaje de modelado: clases, generalizaciones, atributos, asociaciones, y operaciones.

Perfiles UML: es un conjunto de **estereotipos**, **restricciones** y **valores etiquetados** *(extensiones de las propiedades de un elemento UML, para añadir mayor especificación al mismo)*. Estos conceptos forman parte del mecanismo de extensión de UML. En la actualidad existen perfiles para CORBA, java, EJB, o C++ lo que permitirá construir PSM's específicos para estas tecnologías.

Object Constraint Lenguaje (OCL): Es un lenguaje de especificación con el cual podemos escribir expresiones sobre modelos , por ejemplo el cuerpo de una

especificación de consulta, invariantes y pre y post condiciones, lo que nos permite hacer mas precisos y completos los modelos. Se utiliza en MDA para especificar valores iniciales en atributos, definir el cuerpo de operaciones de consulta, especificar reglas de derivación para atributos y asociaciones y expresar restricciones sobre clases y atributos.

Aportes y Beneficios de MDA

Aportes metodológicos

- ✓ Los **modelos** dejan de ser utilizados únicamente para la documentación o como guía de implementación, convirtiéndose en el **artefacto principal** del proceso de desarrollo.
- ✓ MDA aporta **lenguajes estándares** que pueden utilizarse para aplicar MDD: UML, MOF, XMI, OCL, CWM y QVT.

- ✓ **Estandarización** del Model Driven Development (MDD).

Aportes y Beneficios de MDA

Aportes cualitativos

✓ **Productividad**: Se puede añadir funcionalidad con menos esfuerzo (menor costo de mantenimiento). El trabajo “sucio” recae sobre las herramientas y no sobre el desarrollador.

✓ **Ventajas en la arquitectura**: la elaboración de un modelo independiente de plataforma primeramente fuerza al diseñador a pensar acerca de la arquitectura y el modelo de objetos del sistema.

✓ **Portabilidad**: el PIM es reusable y puede obtenerse a partir de este modelo el código de la aplicación para varias plataformas de destino.

Aportes y Beneficios de MDA

Aportes cualitativos

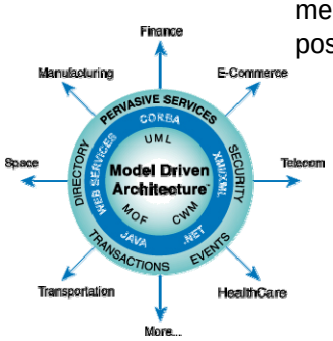
✓ **Interoperabilidad**: No sólo se contempla la generación de las capas de aplicación (lógica, BD, GUI) sino además los “puentes” o “pasarelas” entre ellos para que puedan interoperar efectivamente.

✓ **Mantenimiento y documentación**: los modelos desarrollados también sirven de documentación del sistema. Permiten fácil comprensión de la arquitectura y funcionamiento del sistema para poder introducir ajustes o cambios debido a nuevos requisitos.

Lenguajes estándares de MDA

UML: Lenguaje estándar unificado para especificar, visualizar, construir y documentar los distintos artefactos de un sistema software.

OCL: Lenguaje declarativo que sirve para definir *constraints* y *queries* (consultas) sobre objetos de cualquier modelo UML, MOF o metamodelo derivado de MOF. También sirve para expresar pre y post- condiciones.



XMI: Estándar de almacenamiento de modelos y metamodelos por medio de XML. De interés para los desarrolladores de herramientas MDA o de modelado que permitan exportar/importar a este formato.

MOF: Lenguaje abstracto para la especificación de lenguajes de modelado, o sea, metamodelos. Se especifica formalmente la sintaxis abstracta de un metamodelo utilizando el conjunto de constructores del modelo de clases orientado a objetos.

CWM: Lenguaje de modelado parecido a UML pero que incorpora metadatos que representan conceptos utilizados en las áreas de datawarehousing y análisis de negocios (datamining, etc.)

QVT: Lenguaje para la definición de transformaciones de modelos. Es una extensión de OCL 2.0 y permite realizar consultas (*queries*) sobre los modelos, definir vistas (*views*) y transformaciones (*transformations*).

Metamodelos y MDA

En MDA es crucial analizar, automatizar y transformar modelos:

- Mantener trazas y relaciones entre diferentes modelos
- Lograr interoperabilidad en diferentes niveles

↓

Descripciones precisas de la semántica de los modelos

↓

Metamodelos

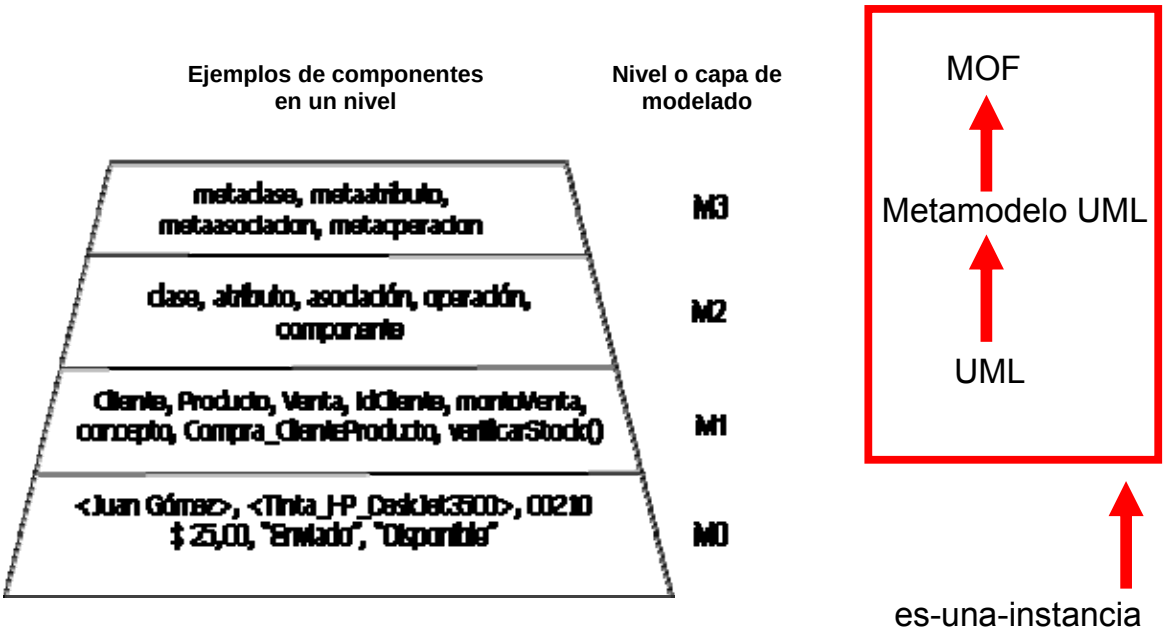
Metamodelos y MDA

Permite definir nuevos metamodelos, que sirvan para representar conceptos y detalles de un dominio específico.

Los constructores de UML para ciertos dominios son insuficientes por ser demasiado genéricos y numerosos.

Para **ordenar los metamodelos** que derivan de MOF, el OMG ha establecido una arquitectura de cuatro niveles...

Arquitectura de metamodelado (4 niveles)



Plataforma en MDA

Plataforma

Es un conjunto de subsistemas y tecnologías que proveen un conjunto coherente de funcionalidad que puede ser usada en cualquier aplicación sin tener en cuenta detalles de cómo la funcionalidad es implementada

Modelos y MDA

Distingue diferentes tipos de modelos:

- CIM (Computation Independent Model)
- PIM (Platform Independent Model)
- PSM (Platform Specific Model)
- ISM (Implementation Specific Model)

MDA – Model Driven Architecture

Computation Independent Model (CIM)

- Es una descripción de la lógica del negocio desde una perspectiva independiente de la computación. Es un modelo del dominio

Platform Independent Model

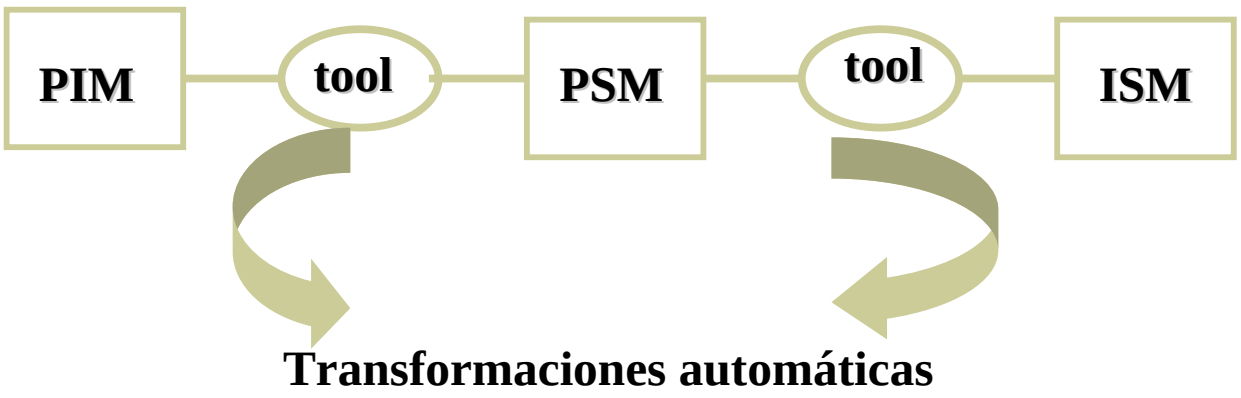
- Es una descripción de la funcionalidad del sistema en forma independiente de las características de plataformas de implementación específicas.

MDA – Model Driven Architecture

Platform Specific Model (PSM)
Es una descripción del sistema en términos de una plataforma específica

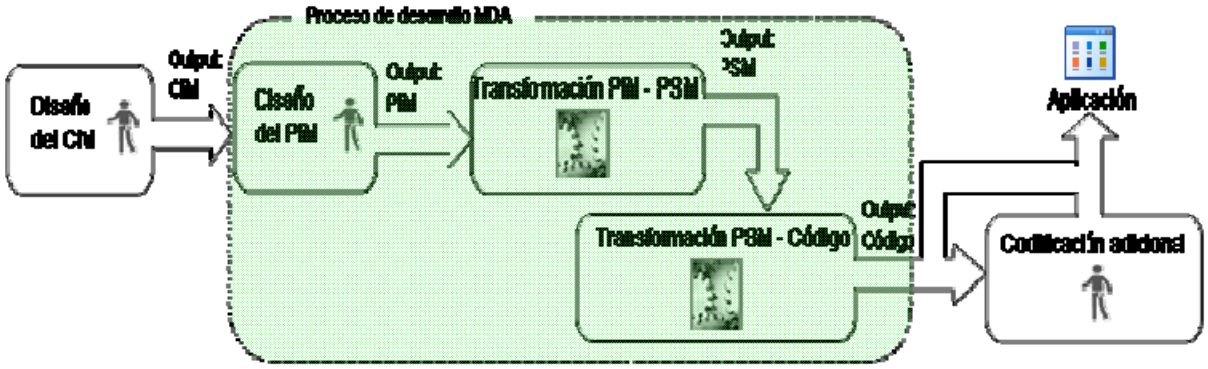
Implementation Specific Model
Es una descripción (especificación) del sistema a nivel de código

Transformaciones en MDA



Transformaciones en MDA

Fases



MDD MODEL DRIVEN DEVELOPMENT

Introducción
Desarrollo Dirigida por Modelos

El Desarrollo Dirigido por Modelos (MDD) se ha convertido en un nuevo paradigma de desarrollo software que promete una mejora de la productividad y de la calidad del software a través de un proceso guiado por modelos y soportado por potentes herramientas que generan código a partir de modelos.

El paradigma MDD tiene dos ejes principales:

- por un lado hace énfasis en la separación entre la especificación de la funcionalidad esencial del sistema y la implementación de dicha funcionalidad usando plataformas tecnológicas específicas. Para ello, el MDD identifica dos tipos principales de modelos: modelos con alto nivel de abstracción e independientes de cualquier tecnología de implementación, llamados PIM (Platform Independent Model) y modelos que especifican el sistema en términos de construcciones de implementación disponibles en alguna tecnología específica, conocidos como PSM (Platform Specific Model);

- por otro lado, los modelos son considerados los conductores primarios en todos los aspectos del desarrollo de software. Un PIM es transformado en uno o mas PSMs, es decir que para cada plataforma tecnológica específica se genera un PSM especifico. La transformación entre modelos constituye el motor del MDD y de esta manera los modelos pasan de ser entidades meramente contemplativas a ser entidades productivas. MDD reduce el salto semántico entre el dominio del problema y de la solución, se reducen los tiempos de desarrollo y las herramientas de generación pueden aplicar framework, patrones y técnicas cuyo éxito se ha comprobado.

La iniciativa MDD cubre un amplio espectro de áreas de investigación: lenguajes para la descripción de modelos, definición de lenguajes de transformación entre modelos, construcción de herramientas de soporte a las distintas tareas involucradas, aplicación de los conceptos en métodos de desarrollo y en dominios específicos, etc. y para esto MDD propone el uso de un conjunto de estándares como MOF, UML y QVT. Actualmente, algunos de estos aspectos están bien fundamentados y se están empezando a aplicar con éxito, otros sin embargo están todavía en proceso de definición.

MDA según algunos Autores

Los ingenieros de software y la comunidad científica han recibido con mucho interés la propuesta MDA. Fruto de este interés está surgiendo numerosos eventos relacionados con el tema y un elevado número de publicaciones que tratan aspectos relacionados con MDA. En un contexto de creciente difusión es complejo realizar un estudio sin dejar a algún autor o documento por estudiar. En este trabajo hemos seleccionado y estudiado una serie de publicaciones que consideramos relevantes sobre MDA. En este apartado se comentan las posturas MDA a Debate 3 que cada uno de los autores mantiene respecto a las

cuestiones anteriormente planteadas. El cuadro 1 presenta un resumen de la visión que ofrece cada autor frente a dichas cuestiones.

Objetivos principales del enfoque MDD

- Mejorar la Calidad del Software
- Mejorar la mantenibilidad del software
- Mejorar niveles de reusabilidad
- Manejo de la complejidad – abstracción
- Productividad
- Interoperabilidad

Naturaleza de MDA

Para empezar con la primera cuestión, la naturaleza de MDA, es interesante acudir a la propia Gula de MDA. **Según la guía:** *MDA is an approach to using* MDA a Debate 5 *Models in software development*. Esto es lo que dicen también Bichler y Czarnecki.Sin embargo, Atkinson denomina de forma genérica Desarrollo de Software Guiado por Modelos (o MDD en inglés) a esta aproximación al desarrollo de software. Esta posición se encuentra reforzada por Mellor y los editores de IEEE

Software, para los que MDA es un conjunto de estándares de OMG que permiten seguir la aproximación que defiende el MDD. Precisamente la intención de convertirse en un estándar es un hecho que también destaca Bichler.

Tomando como base las interpretaciones anteriores, podemos determinar que: **Model Driven Development (MDD)** es una **aproximación al desarrollo de software** basado en el modelado del sistema software y su generación a partir de ellos. Al ser únicamente una aproximación, sólo proporciona una estrategia general a seguir en el desarrollo de software, pero no define ni técnicas a utilizar, ni fases del proceso, ni ningún tipo de guía metodológica. **Model Driven Architecture (MDA)** es un **estándar de OMG que defiende el MDD y agrupa varios lenguajes que pueden usarse para seguir esta aproximación**. Según esta afirmación, MDA posee el valor añadido de proporcionar lenguajes con los que definir métodos que sigan MDD. Sin embargo, MDA no es por sí mismo un método que define técnicas, etapas, artefactos, etc., solamente proporciona la infraestructura tecnológica y conceptual con la que construir estos métodos MDD.

En definitiva, **MDA no puede ser utilizado directamente para desarrollar software**, es necesario que se construyan métodos precisos que proporcionen a los equipos de desarrollo pautas y técnicas que éstos puedan seguir y utilizar.

Novedades de MDA
Una vez aclarada la primera cuestión pasamos a tratar el segundo aspecto. Según las posiciones de los autores, cabría distinguir entre aquellas aportaciones o méritos propios del MDD en general y aquellas que corresponden únicamente a MDA.

Aportaciones de MDD. Todos los autores están de acuerdo en que el aspecto novedoso de MDD (y, por tanto, de MDA en concreto) respecto a los métodos tradicionales de desarrollo de software que usan modelos es que **los modelos dejan de ser utilizados únicamente para la documentación o como guía para la implementación, convirtiéndose en el artefacto principal del proceso de desarrollo**. Una vez especificado el sistema, los modelos se transforman en el producto final.

Como indican explícitamente Selic y Gardner, la clave para que sea posible llevar a cabo la filosofía de desarrollo promulgada por MDD es la *transformación de los modelos*. Hasta ahora esta etapa era ignorada o tratada de forma escueta e informal por los métodos que utilizaban modelos.

Aportaciones de MDA. Los autores seleccionados asignan diversas carácter ísticas novedosas a la propuesta MDA, sin embargo en este caso no existe un consenso sobre todas ellas. Debido a esta situación, hemos ordenado las aportaciones de mayor a menor nivel de seguridad de que realmente se trata de una novedad. **aporta lenguajes estándares** que pueden utilizarse para aplicar MDD. Es curioso observar como, pese a que se está trabajando en ello en la petición de propuestas QVT (Query View Transformation) [16], todavía no existe un estándar para la especificación de las transformaciones entre modelos, que constituyen un elemento clave de esta propuesta. **pretende estandarizar el MDD** dentro de la comunidad de la Ingeniería del Software. Si esto llegase a ocurrir, se trataría de una revolución importante o paso de gigante, porque supondría pasar de métodos de desarrollo basados en lenguajes de programación (principalmente imperativos) a métodos basados en modelos conceptuales.

introduce los modelos de diseño e implementación (PSM). Esta aportación es considerada por Agrawal. Se trata de una afirmación discutible. Si bien es cierto que muchos métodos tradicionales utilizan modelos conceptuales del sistema construidos mediante primitivas de alto nivel de abstracción, en otros métodos y, sobre todo, en proyectos reales, los modelos se utilizan principalmente como representaciones directas de las entidades del software en desarrollo (clases y tipos de datos de algún lenguaje de programación concreto, referencias entre clases, etc.). Por lo tanto, no se puede decir que el uso de modelos de diseño sea una aportación de MDA.

Grado de Automatización de las Transformaciones La tercera cuestión que se pretende discutir en este trabajo gira alrededor el grado de automatización que deben tener las transformaciones entre modelos.

La **Gula de MDA** describe tres opciones: que las transformaciones sean realizadas de manera *completamente manual* por parte de los desarrolladores del sistema. que sea necesaria la participación de los desarrolladores para llevar a cabo la transformación. Siguiendo esta estrategia, la transformación puede estar *guiada por herramientas* o puede aplicarse automáticamente para generar automáticamente una versión preliminar de los modelos transformados. que a partir de modelos independientes de plataforma sea posible obtener de manera *completamente automática* el sistema. Según estas opciones no es necesario que un método o herramienta especifique transformaciones completamente automáticas para que pueda decirse que sigue el estándar MDA. Frente a esta postura, la mayoría de autores consideran que la clave del éxito del MDD reside en que las transformaciones de los modelos al código fuente del sistema se realice de manera completamente automática. Algunos, como Selic MDA a Debate y Weis, incluso exigen que se cumpla esta característica en el MDD. En caso contrario, como indica Weis, los desarrolladores pueden considerar que realizar el modelado del sistema y la transformación manual es demasiado costosa frente a la opción de desarrollar el sistema directamente utilizando la tecnología destino. En resumen, respecto a esta cuestión

podemos concluir que:
1. **no es necesario automatizar las transformaciones** para seguir el estándar
MDA , pero

2. una parte importante de la comunidad de la Ingeniería del Software considera que **es necesario automatizar las transformaciones para que el MDD sea realmente útil**

MDD – Model Driven Development

- Automatización de las transformaciones
  - Según la guía del MDA las transformaciones pueden ser:
    - Completamente manuales
    - Semiautomáticas o asistidas
    - Completamente automáticas
  - Para muchos autores la automatización es clave en MDD
  - En resumen:
    - No es imprescindible automatizar las transformaciones para seguir el estándar MDA
    - Sí es necesario automatizar las transformaciones para que MDD sea útil

Bibliografía
MDA & MDD
http://es.wikipedia.org/wiki/Model_Driven_Architecture
http://ajayu.memi.umss.edu.bo/naverus/weblog/arquitectura-dirigida-modelos-mda
http://personal.telefonica.terra.es/web/lencorredera/mda_j2me.pdf
http://www.sparxsystems.com.ar/platforms/mda_tool.html
http://sol.info.unlp.edu.ar/eclipse/MDA/index.html
http://www.dsic.upv.es/workshops/dsdm04/files/Actas-DSDM04.pdf

http://sol.info.unlp.edu.ar/eclipse/MDA/index.html
http://www.lcc.uma.es/~av/MDD-MDA/

Mellor S., Scott K., Uhl A., Weise D. MDA Distilled: Principles of Model-Driven Architecture. Editorial Addison-Wesley. 2004

Especificaciones de MDA. Disponible en http://www.omg.org/cgi-bin/doc?omg/03-06-01 y http://www.omg.org/cgi-bin/doc?ormsc/05-04-01