

[white paper]

Arquitectura dirigida por modelos

En el presente artículo veremos qué es Model Driven Architecture (MDA) y de qué manera influye en los desarrollos basados en ella.

Actualmente, MDA es importante para el desarrollador promedio. Quienes fuimos bendecidos con la capacidad de utilizar la creatividad para transformar ceros y unos en aplicaciones útiles para nuestros clientes, sabemos que debemos estar al tanto de lo que ellos puedan necesitar.

Es una realidad que, en los últimos dos años, muchas organizaciones han comenzado a prestar atención a MDA, ya que promueve el uso eficiente de modelos de sistemas en el proceso de desarrollo de software.

MDA representa para nosotros, los desarrolladores, una nueva manera de organizar y de administrar arquitecturas empresariales, basada en el uso de herramientas de automatización de etapas en el ciclo de desarrollo y servicios, para definir los modelos y facilitar transformaciones paulatinas entre ellos. Algunos ejemplos de modelos son el de análisis, el de diseño y el de comportamiento, entre otros. Es decir que, a partir de uno de ellos, podemos generar otro de menor abstracción.

Un ejemplo común de generación de modelos es la generación de código a partir del modelo de diseño, mediante el uso de una herramienta UML. En este caso, usamos una herramienta para transformar el modelo de diseño en el "modelo" de código.

¿Qué es MDA?

MDA es el acrónimo de *Model Driven Architecture* (arquitectura dirigida por modelos), un concepto promovido (pero no creado) por la OMG, que propone basar el desarrollo de software en modelos especificados utilizando UML para que, a partir de ellos, se realicen transformaciones que generen código u otro modelo, con características de una tecnología particular (o con menor nivel de abstracción).

Suele escucharse decir que MDA es la evolución natural de UML, ya que tiende a incrementar la cantidad de código generado, a partir de especificaciones detalladas en UML.

Modelos MDA

Los modelos juegan un rol trascendental en MDA. Como un framework para construir sistemas, MDA abstrae el sistema por construir en distintas capas de abstracción (*layers*). Tradicionalmente, el OOAD (*Object Oriented Analysis and Design*) contiene, entre otros, una vista de análisis, una vista de diseño detallado y código representando la vista de negocios de un sistema, la vista de arquitectura y la vista de implementación. MDA agrega una capa de abstracción más, que representa el contexto de negocio del sistema. En la [Figura 1] se muestran las diferentes capas de abstracción. El gráfico se hace más abstracto hacia la izquierda y se vuelve más concreto hacia la derecha.

¿Qué no es MDA?

Hoy, MDA es uno de los tantos acrónimos de moda y, como ocurre algunas veces con las modas, el concepto puede tender a malinterpretarse. Por lo tanto, enumeremos rápidamente qué no es MDA.

MDA no es un proceso de desarrollo.

MDA no es una especificación.

MDA no es una implementación.

MDA no es una implementación de referencia de ningún estándar particular.

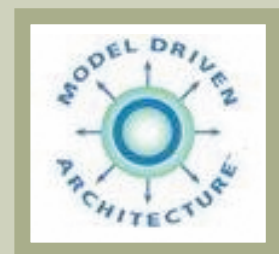
MDA no es un concepto maduro aún.

MDA no es, simplemente, generar código.

MDA no tiene, aún, una visión unificada en la industria.

MDA no es una arquitectura ni un "architectural style o pattern".

MDA es sólo un concepto; representa una nueva manera de organizar y de administrar arquitecturas empresariales. Las herramientas que proponen soluciones MDA son aquellas que facilitan la implementación de este concepto.



Aquí podemos ver el logo que identifica a MDA.

Valerio Adrián Anacleto

Socio Fundador de Epidata
Consulting S.R.L
adrian@epidataconsulting.com



Los modelos concretos exceden en número a los abstractos. A medida que avanzamos en las transformaciones, los modelos se vuelven más concretos, transformando al modelo abstracto en uno compatible con una tecnología o plataforma. La situación inversa de llevar el código hacia un modelo concreto –también conocida como ingeniería reversa– rara vez ocurre, excepto cuando el punto de partida es el código mismo.

Esto se produce debido a que MDA promueve la fuerte separación entre las responsabilidades de requerimientos del negocio y las responsabilidades tecnológicas. La ventaja de esta “separación de responsabilidades” es que ambos aspectos pueden evolucionar independientemente sin generar dependencias entre sí. De esta manera, la lógica de negocio responderá a las necesidades del negocio y no, a las vicisitudes técnicas.

CIM

El CIM (*Computational-Independent Model*) se centra en los requerimientos y representa el nivel más alto del modelo de negocios. Usa un lenguaje para modelar procesos de negocios que no es UML, aunque éste puede ser derivado perfectamente utilizando MOF (*Meta-Object Facility*) (para conocer más detalles sobre MOF y UML, ver **code** #22 y #25).

El CIM trasciende los sistemas. Cada proceso de negocio interactúa con trabajadores humanos y/o componentes de máquina. CIM describe solamente aquellas interacciones

que tienen lugar entre los procesos y las responsabilidades de cada trabajador, sea o no humano.

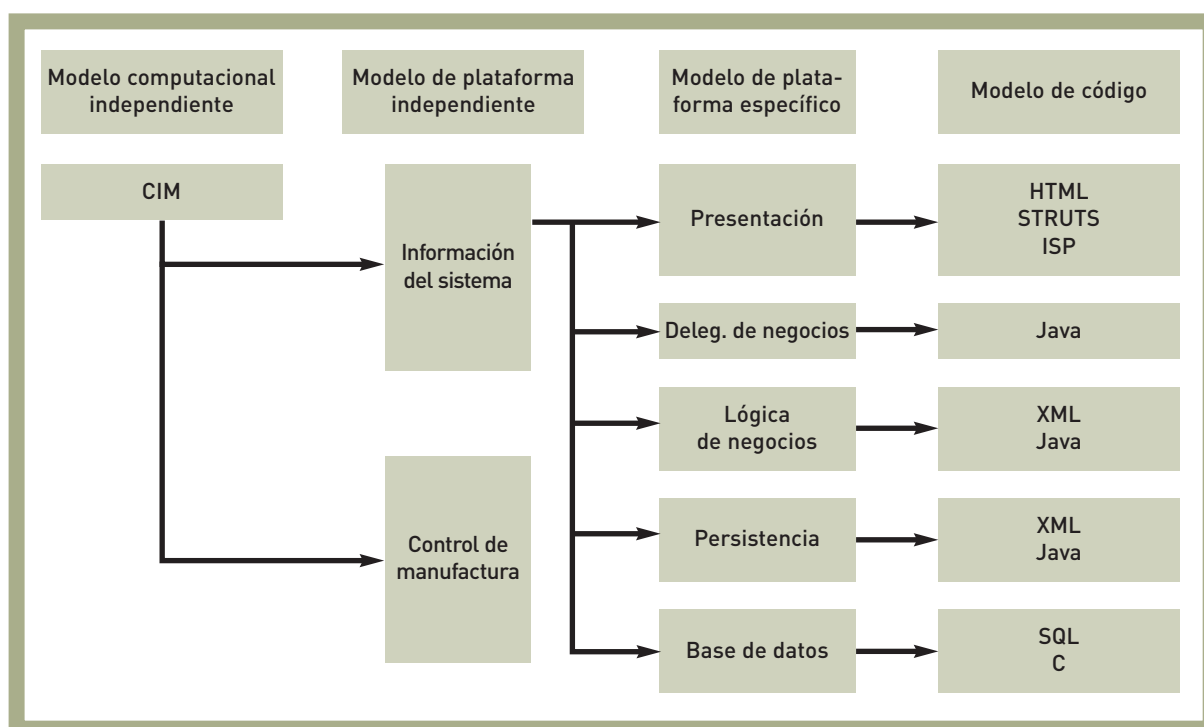
Un objetivo fundamental de CIM es que cualquiera que pueda comprender el negocio y sus procesos pueda comprender un CIM, ya que éste evita todo tipo de conocimiento especializado o de sistemas. A este foco en el negocio por sobre la tecnología debe su nombre, que en español se traduce como Modelo Independiente de la Computación.

MDA define un framework para procesar y relacionar modelos.

PIM

El PIM (*Platform-Independent Model*), que se traduce al castellano como Modelo Independiente de la Plataforma, representa el modelo de procesos de negocio que va a ser implementado. Comúnmente se usa UML o un derivado de él para describir el PIM.

El PIM modela los procesos y la estructuras del sistema, sin hacer ninguna referencia a la plataforma en la (o las)



[Figura 1] Un ejemplo de modelo MDA y sus relaciones.

[white paper - Arquitectura dirigida por modelos]

Recursos más importantes

www.theserverside.com/tt/articles/article.tss?l=MDA_Haywood
Interesante link con críticas a MDA.

en.wikipedia.org/wiki/Model-driven_architecture#MDA_tools
Herramientas MDA.

www.omg.org/mda
Sitio oficial de la OMG dedicado a MDA.

www.omg.org/docs/omg/03-06-01.pdf
Guía oficial de la versión 1.0.1 de MDA.

www.lcc.uma.es/~av/MDD-MDA/#MDA%20y%20estandares
Lista de distribución y muchos recursos en español

planetmde.org
Uno de los mejores sitios como puntero de recursos MDE, que incluye herramientas, libros y cursos en universidades, entre otros.

que será desplegada la aplicación. Ignora los sistemas operativos, los lenguajes de programación, el hardware y la topología de red.

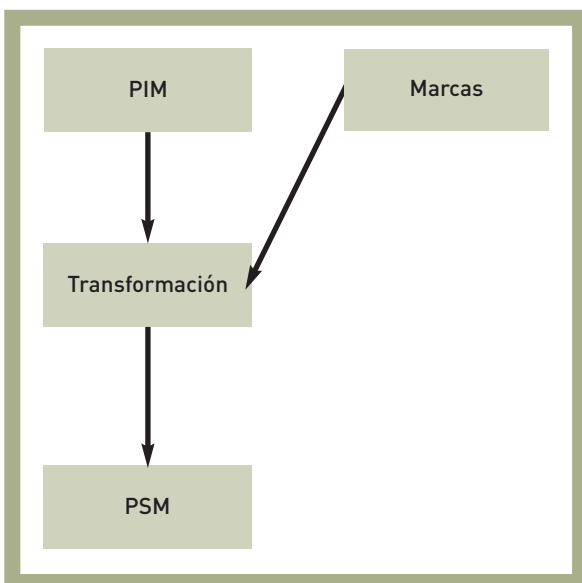
Suele ser el punto de entrada de todas las herramientas para MDA e, incluso, de muchos artículos que hablan de MDA, y dejan de lado CIM.

PSM

Los PSMs tienen que lidiar explícitamente con los sistemas operativos, los lenguajes de programación, las plataformas (CORBA, .NET, Java EE), etc.

Code model

El modelo de código representa el código desplegable (*deployable*), normalmente, en un lenguaje de programación de alto nivel, como Java, C#, C++, VB, JSP, y otros.



[Figura 2] Los PIMs se transforman en PSMs.

Idealmente, el modelo de código está listo para compilar y no debería requerir la intervención humana; el despliegue de la aplicación podría ser automatizado.

Según los puristas y algunos fanáticos de MDA, en un ambiente MDA maduro, no deberíamos pensar en el código más que como simples archivos o como un mero objeto intermedio para generar el ejecutable final.

Pero debido a que MDA no está maduro y difícilmente se llegue alguna vez a la utopía de no tener que tocar ningún código, los desarrolladores seguiremos necesitando conocer la tecnología para complementar la generación de código, debuggear la aplicación y, sobre todo, lidiar con muchos y variados errores inesperados, extraños y divertidos.

Decisiones de diseño

Hemos visto cómo, de manera automática y paulatina, MDA promueve la transformación de modelos que representan lógicas de negocios complejas (CIM), hasta llegar al código ejecutable y desplegable (*code model*). Pero ¿qué pasa con las decisiones de diseño, aquellas que tomamos cuando, por ejemplo, tenemos que desarrollar un sistema en el cual la mayoría de las transacciones sólo leen datos, pero diez de ellas hacen uso intensivo del procesador?

¿Qué sucedería si todos los mapeos se hicieran de igual manera? Nos es dado pensar que esto degradaría la calidad final percibida de la aplicación. Para corregir este problema, MDA promueve el uso de *marks* (marcas), que indican aspectos específicos para tener en cuenta en cada transformación.

Un PIM generado a partir de un PIM y marcas se denomina PIM marcado o *Marked PIM*. El uso de las marcas establece que las decisiones referidas a aspectos tecnológicos queden fuera de los modelos principales.

Ahora bien, para realizar el mapeo entre un PIM marcado y, por ejemplo, un PSM, es necesario detallar cómo se mapean esas marcas, para lo cual se definen los *mappers* (mapeadores). Puede verse la relación entre los mapeadores, las marcas y los PSM en la [Figura 3]. Los mapeos se hacen utilizando un QVT: *Query, Views, and Transformations*.

Ventajas de MDA

La ventaja principal de MDA radica en una clara y estricta separación de responsabilidades. Por un lado, modelaremos los PIMs, que representan los modelos de nuestro negocio, y por el otro, los PSMs con las preocupaciones tecnológicas. Esto permitirá que ambos modelos evolucionen por separado. De esta manera, si quisiéramos, por ejemplo, modificar un aspecto técnico, bastaría con modificar el PSM sin que éstos tuvieran impacto en la lógica de negocios.

Esta idea parte de un concepto que, en ingeniería de software, se llama *Guías de Diseño*. Particularmente, una de esas guías dice que el modelado de la solución debe ser guiado por el negocio. Esta guía se basa en la afirmación de que un cambio en el negocio seguramente producirá uno en el código, pero que los cambios en el código no deberían impactar en el negocio.

MDA también permite lidiar con la complejidad del negocio, modelando a éste por separado, y permitiendo su análisis y

¿Qué es MDD?

Junto a MDA, ésta es otra sigla que suena mucho últimamente. MDD es el acrónimo de Model Driven Development (Desarrollo Dirigido por Modelos), y concibe al desarrollo de software bajo la idea central de que los artefactos fundamentales son los modelos (y no, los programas). Implica la generación automática de programas a partir de modelos. En MDD, "el modelo es la implementación"

MDA no es más que un MDD particular, con nombre propio (y estándar definido por la OMG).

mejora; disminuir costos, si se cuenta con una herramienta MDA adecuada a nuestras necesidades; y mejorar la calidad de nuestros modelos y procesos, mediante su análisis y la separación de responsabilidades.

Estándares involucrados en MDA

Las tecnologías más importantes involucradas para poder llevar a la práctica los conceptos subyacentes en MDA son: *Meta Object Facility* (MOF), *Unified Modeling Language* (UML), *XML Model Interchange* (XMI), *Common Warehouse Meta-model* (CWM), *Software Process Engineering Meta-model* (SPEM), *Action Semantics Language* (ASL), *Query-View-Transformation* (QVT) y *UML profiles*.

Para leer un resumen de ellas, se pueden consultar las ediciones de *.code* #22 y #25, en las cuales hablamos de UML 2.0, explicamos la nueva estructura de UML, y cómo las tecnologías aquí nombradas permiten hacer de UML un lenguaje extensible y orientado a la obtención de modelos ejecutables.

Herramientas MDA

Las herramientas MDA deberían de proveer la capacidad de transformar modelos de negocios puros (CIMS) en aplicaciones completas, desplegables y capaces de ejecutar con un mínimo de decisiones técnicas. Lamentablemente, nos encontramos muy lejos, por ahora, de que MDA o alguna herramienta MDA brinde todas estas facilidades para cual-

quier negocio y problemática que se vaya a desarrollar. Tampoco existe un líder en herramientas MDA, debido a que son herramientas muy sofisticadas. Algunas de las más conocidas con:

- ATL ATLAS Transformation Language
- OptimalJ is a MDA tool for J2EE
- ArcStyler is a MDA tool for J2EE and .NET
- UMT UML Model Transformation
- ArgoUML
- Codagen
- Rational Architect
- MDA Transf
- Enterprise Architect
- GReAT
- AndroMDA

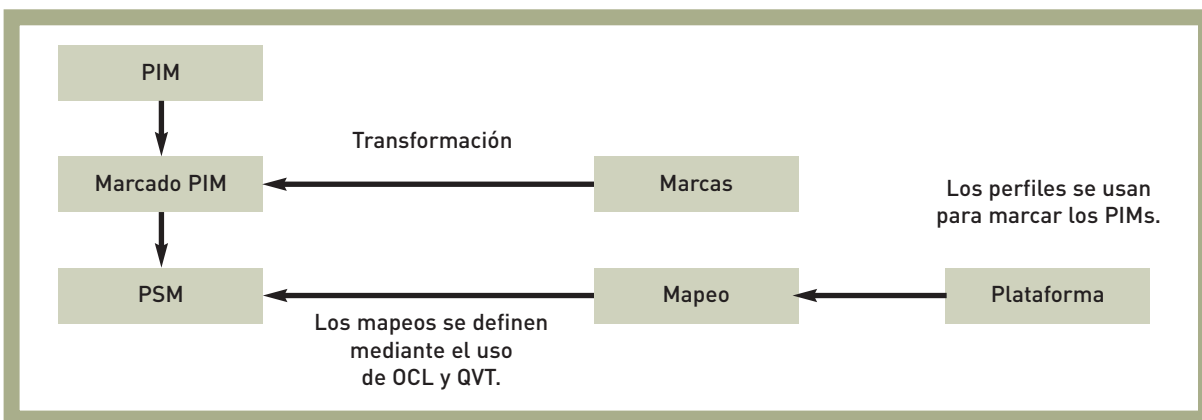
MDD vs. MDE

Por ser MDD una marca registrada de la OMG, la comunidad científica utiliza el término MDE para referirse a ideas y conceptos relacionados con la ingeniería basada en modelos, sin centrarse, exclusivamente, en los estándares de la OMG.

En definitiva, MDD significa lo mismo que MDE, sólo que MDD es una marca registrada, mientras que MDE no. Si habláramos de medicamentos, MDE sería el nombre genérico de la droga, mientras que MDD sería una marca comercial, cuyo componente principal es la droga MDE.

¿Cómo evaluar una herramienta MDA?

Con el advenimiento de las palabras de moda, utilizadas en exceso y en muchos casos de manera inapropiada (*buzzwords*), llegan a la puerta de nuestros gerentes los vendedores de sueños y espejitos de colores, comentando que, con la nueva herramienta MDA, podremos resolver todos nuestros problemas sólo presionando el botón derecho del mouse. Como todo el



[Figura 3] Uso de marcas y mapeos.

[white paper - Arquitectura dirigida por modelos]

mundo sabe, ese tipo de expectativas suele ser exagerada. En mi opinión personal, la forma de probar una herramienta es usarla para resolver nuestra problemática y, si hay vendedores involucrados, pedirles gentilmente que nos ayuden a resolver el problema de nuestro negocio con un caso "testigo" elegido por nosotros. Si la herramienta realmente funciona, el caso tendrá solución, y nosotros seremos unos clientes felices.

Más allá del caso práctico sugerido, debemos tener en cuenta varios factores al momento de la evaluación. Éstos tienen que ver con:

- **El mantenimiento de la aplicación:** ¿Cuánto me va a costar agregar o mantener código una vez que haya pasado por el proceso de generación?
- **Impacto del cambio:** ¿Cómo se administra un cambio en el proceso de negocio? ¿Qué costo tiene?
- **Manejo de excepciones:** ¿Cómo se administran los casos excepcionales, como las transacciones que hacen uso intensivo del procesador?
- **Curva de aprendizaje:** ¿Cuál es la curva de aprendizaje? ¿Tengo que aprender todo un nuevo meta-lenguaje? ¿Basta con el uso intuitivo de herramientas "visuales"?
- **Madurez:** ¿Qué tan maduros son la herramienta y los frameworks involucrados? ¿Cuántos casos de éxito existen y cuáles son?

Como éstas, hay muchas preguntas que debemos hacernos antes de elegir una herramienta, con la cual tendremos que convivir un tiempo determinado y que podría llegar a transformarse en un gasto, para dejar de ser la inversión esperada.

¿Cómo puedo seguir?

En el recuadro "Recursos más importantes" dejamos algunos recursos para quienes busquen interiorizarse más en el tema de MDAs. Por supuesto, serán muy bien recibidos los mails con consultas, sugerencias, experiencias y opiniones.

Conclusión

MDA data del año 2000, cuando la OMG publicó un White Paper titulado "Model Driven Architecture", en el que describía la visión del desarrollo de software a través de modelos de objetos relacionados entre sí, para la generación de sistemas completos.

Si bien transcurrieron casi seis años desde entonces, MDA no es, ni será, una bala de plata, capaz de aniquilar todas las problemáticas inherentes al desarrollo de software.

MDE

MDE es el acrónimo de Model Driven Engineering (Ingeniería Dirigida por Modelos) y hace referencia al uso sistemático de modelos, como los elementos principales en la ingeniería de software, durante el ciclo de vida completo del proyecto.

Hoy en día, a pesar de todo, los modelos son costosos de construir y, una vez construidos, éstos deben ser transformados manualmente en código. Esta tarea es tediosa, propensa a errores, repetitiva en muchos casos y, sobre todo, un proceso caro en recursos (y dinero, por supuesto). Además, una vez que el trabajo interesante y de mayor abstracción fue realizado, sólo la transformación desde el código al ejecutable es automatizable.

MDA también es el resultado de reconocer que la interoperabilidad es algo bueno y que el modelado también lo es. Bien utilizado y teniendo en cuenta los principios de diseño subyacentes, puede ahorrarnos la escritura y la generación de muchas tediosas líneas de código, y esto es siempre bien recibido. Incluso, es probable que, analizando las tecnologías usadas hoy en nuestra organización, encontremos que algunas de ellas están fuertemente relacionadas con el concepto MDA. En ese caso, es posible organizarlas de manera que reflejen y hagan evidente el uso de MDA como concepto subyacente, más allá de las herramientas. ●

Valerio Adrián Anacleto es socio fundador de Epidata Consulting, una empresa especializada en brindar servicios relacionados con la arquitectura de software, como integración de aplicaciones corporativas, validación, definición, implementación de arquitecturas y procesos, mentoring y capacitación. En Epidata se desempeña como consultor y capacitador en el área arquitectura de software. También es Licenciado en Ciencias de la Computación de la Facultad de Ciencias Exactas y Naturales de la U.B.A. (Argentina), donde fue docente durante los últimos nueve años. Es frecuente disertante en actividades de divulgación relacionadas con la ingeniería de software y ha escrito varios artículos sobre el tema.

¿Qué es la OMG?

La OMG (Object Management Group) es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas, de la industria del software, como IBM, Apple Computer, Sun Microsystems Inc y Hewlett-Packard. Esta asociación se encarga de definir y mantener estándares para aplicaciones de la industria de la computación. Otro de los estándares definidos por la OMG, además de UML, es CORBA, que brinda interoperabilidad multiplataforma a nivel de objetos de negocio.

Podemos encontrar más información sobre la OMG en su sitio oficial: www.omg.org.