COMPARATIVO DE HERRAMIENTAS MDA (ANDROMDA, ARCSTYLER, OPTIMALJ)

Cuesta M. Albeiro [*]
López T. Marcelo [**]
Joyanes A. Luis [***]

Resumen

Con el entorno y dinámicas competitivas de la actualidad, contar con tecnología de información y comunicaciones no supone por sí misma una ventaja competitiva para las organizaciones. Es la gestión de esa tecnología la que puede dar una ventaja o marcar factor diferencial para el éxito de estas. De acuerdo a esto, apropiarse de un modelo de gobierno IT, para esta gestión, es un elemento clave para el cumplimiento de los objetivos de la empresa.

Palabras clave: COBIT, Gerencia de TIC, Gestión de TIC, Gobierno IT, ITIL, Modelos de gobierno IT, Serie ISO 27000.

MDA TOOLS COMPARISON (ANDROMDA, ARCSTYLER, OPTIMALJ)

Abstract

With the current surroundings and competitive dynamics, to count on information technology and communications does not, by itself, make up a competitive advantage for organizations. The management of this technology can give an advantage or make the difference for the organization's success. According to this, taking control of an IT governance model, is a key element for the fulfillment of the company's objectives.

Key words: COBIT, ITGovernance, IT Management, ITIL, IT governance models, ISO 27000 Series.

[*] Investigador Grupo de Ingeniería del Software de la Universidad Autónoma de Manizales; Ingeniero de Sistemas; Magíster en Calidad de Software; Doctor(c) en Ingeniería del Software de la Universidad Pontificia de Salamanca en Madrid. E-mail: albeirocuesta@hotmail.com
[**] Profesor Asociado de la Facultad de Ingeniería de la Universidad de Caldas, Colombia; Ingeniero de Sistemas; Magíster en Educación; Magíster en Gestión del Conocimiento; Doctor(c) en Sociedad de la Información y el Conocimiento de la Universidad Pontificia de Salamanca en Madrid. E-mail: mlopez@ucaldas.edu.co.
[***] Catedrático del Departamento de Lenguajes e Ingeniería del Software de la Universidad Pontificia de Salamanca; Doctor en Ingeniería Informática; Doctor en Sociología. E-mail: luis.joyanes@upsam.net

Recibido 2 Septiembre 2009, Aprobado 23 Octubre 2009

Introducción

La construcción de *software* se enfrenta a continuos cambios; actualmente el medio se enfrenta a una nueva forma de creación de *software* en la que los modelos guían todo el proceso de desarrollo de *software*. Guiando el desarrollo con los modelos del *software*, se obtienen beneficios en aspectos como la productividad, la portabilidad, la interoperabilidad y el mantenimiento.

Es así como en el año 2001, el *OMG* (*Object Management Group*) propuso un estándar de trabajo denominado *MDA* (*Model Driven Architecture*), el cual tiene como objetivo acelerar el desarrollo de aplicaciones, simplificar la integración entre distintas tecnologías y reducir el costo de la migración de aplicaciones a nuevas plataformas. La Figura No. 1 ilustra la Arquitectura Conducida por Modelos de *OMG.*

La clave del *MDA* es la importancia de los modelos en el proceso de desarrollo de *software*. *MDA* propone la definición y uso de modelos a diferentes niveles de abstracción para "guiar" todo el proceso de desarrollo (análisis, diseño, mantenimiento y hasta la integración), así mismo da la posibilidad de la generación automática de código a partir de los modelos definidos y de las reglas de transformación entre dichos modelos.

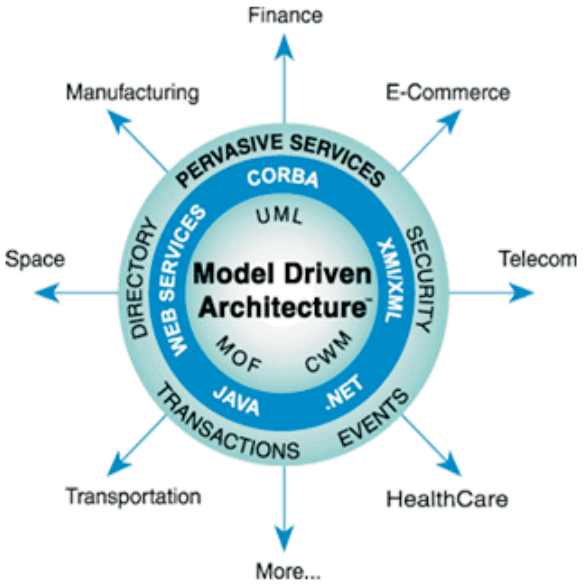

Figura No. 1. OMG: Model Driven Architecture.
Fuente: www.omg.org

Conceptos

El Ciclo de Vida del Desarrollo al utilizar *MDA* consta de un modelo independiente de la plataforma (*PIM*), un modelo específico a la plataforma (*PSM*) y un Código.

MDA plantea que del análisis de requisitos se obtiene un Modelo Independiente de la Plataforma (*PIM*), posteriormente este modelo se transforma con la ayuda de herramientas en uno o varios Modelos Específicos de la Plataforma (*PSM*), y por último los *PSM* se transforman en código.

La Figura No. 2 ilustra la transformación del *PIM* en un *PSM* y de éste en Código.

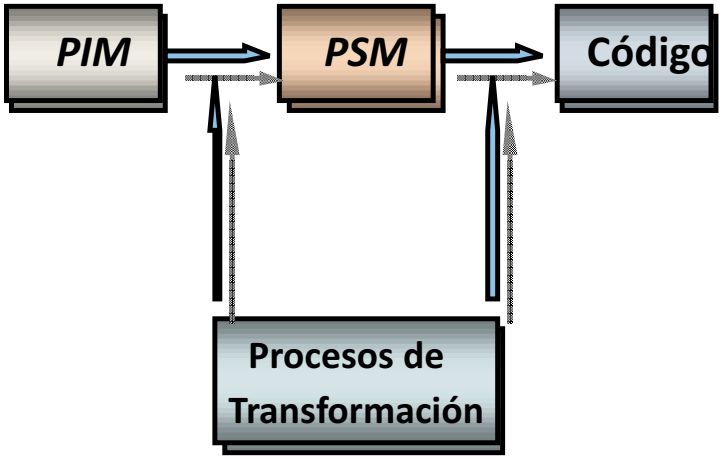

Figura No. 2. Proceso de transformación en MDA.

PIM: es un modelo de alto nivel del sistema independiente de cualquier tecnología o plataforma que permite una abstracción de las características técnicas específicas de las plataformas de despliegue. Muestra aquellas partes de la especificación del sistema que no cambian de una plataforma a otra y es el único que debe ser creado íntegramente por el desarrollador. A partir de un *PIM* pueden generarse varios *PSM*, cada uno definiendo el sistema desde puntos de vista diferentes.

PSM: es un modelo de más bajo nivel que el *PIM* que describe el sistema de acuerdo con una tecnología de implementación determinada, surge a partir de las transformaciones aplicadas sobre el *PIM* obteniendo así 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.

Código: la generación de código se lleva a cabo a partir de cada *PSM* y dado a que éste está muy ligado a una tecnología concreta, la transformación puede ser automática.

En ese orden de ideas los cambios en los procedimientos de negocio o la incorporación de nuevas funcionalidades pueden hacerse de una manera más simple sin tener que hacer cambios en todos los niveles del proyecto. Simplemente se desarrollan los cambios en el *PIM*, y éstos se reflejarán en toda la aplicación, consiguiendo una disminución de trabajo en el equipo de desarrollo, una reducción en los costos del proyecto y por ende se aumenta la productividad.

MDA se apoya sobre los siguientes estándares para llevar a cabo su función:

- *UML*: lenguaje de modelado adoptado por *MDA*, empleado para la definición de los *PIM* y los *PSM*. Es un estándar para el modelado introducido por el *OMG*; cabe mencionar que los Modelos de Clases son los más importantes dentro de *MDA*, ya que el *PIM* y la mayoría de los *PSM* se representan mediante Diagramas de Clases de *UML*.
- *MOF*: establece un marco común de trabajo para las especificaciones del *OMG*, a la vez que provee de un repositorio de modelos y metamodelos. Mediante *MOF* puede definirse cualquier lenguaje de modelado, incluido *UML*.
- *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*

Con el fin de entender *MDA* y sus características, su funcionamiento y su aplicación al proceso de desarrollo, a continuación se revisarán 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 (Mellor, Stephen J et al., 2004). 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: 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 (Frankel David S, 2003).

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.

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 cómo 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 (Kleppe Anneke et al., 2003; Arlow Jim, Neustadt Ila, 2004).

MDA-Cartridges: un Cartucho *MDA* o *MDA*-Cartridge contiene las reglas necesarias para realizar una transformación de modelos. Pueden ser instalados como *plugin*, descargarse de Internet, y editarse o extenderse si es necesario.

Herramientas a comparar

AndroMDA: es un sistema basado en cartuchos, que admite como entrada descripciones *XMI* de diagramas *UML*, y usa XDoclet como tecnología de marcado para el acceso a datos desde las clases Java. Admite como entrada ficheros *XMI* versión 1.1, y como herramienta de modelado la comunidad de desarrollo aconseja el uso de Poseidon for *UML*, de Gentleware.

Admite cualquier lenguaje de programación como salida, y admite código propio para la generación de código.

ArcStyler: es un sistema basado en uso de cartuchos para descripción de transformaciones que permite generar aplicaciones de n capas codificadas en java/J2EE y c#/.NET a partir de diagramas *UML* y la especificación de los procesos del negocio. Permite extender las capacidades de transformación, generando nuevos cartuchos a partir de *UML*, cuyo objetivo sea cualquier plataforma o lenguaje.

No soporta diagramas de componentes ni diagramas de despliegue, pero admite código propio para la generación de código. ArcStyler de *iO-Software* es una herramienta *MDA* que también utiliza *MOF* para soportar estándares como *UML* y *XMI*, y además JMI para el acceso al repositorio de modelos. Integra herramientas de modelado (*UML*) y desarrollo (ingeniería inversa, explorador de modelos basado en *MOF*, construcción y despliegue) con la arquitectura CARAT que permite la creación, edición y mantenimiento de *cartuchos MDA* (*MDA-Cartridge*) que definen transformaciones. También incluye herramientas relacionadas con el modelado del negocio y el modelado de requisitos por lo que cubre todo el ciclo de vida (Warmer Jos, Kleppe Anneke, 2003).

OptimalJ: este producto de la compañía *Compuware* genera aplicaciones J2EE partiendo de los modelos. Implementa completamente la especificación *MDA*. Está desarrollado en Java, lo que le hace portable a cualquier plataforma para su ejecución (Corredera de Colsa Luis Enrique, 2007).

Admite *XMI* versión 1.1 tanto para la importación de ficheros como para su salida. OptimalJ es una herramienta *MDA* que utiliza *MOF* para soportar estándares como *UML* y *XMI*. Se trata de un entorno de desarrollo que permite generar aplicaciones J2EE completas a partir de un *PIM*.

Del proceso de desarrollo con OptimalJ se puede destacar:

• Generación automática a partir del *PIM* de los modelos *PSM* de la capa de presentación (web), capa de negocio EJB y base de datos, estableciendo la conexión (puentes) entre las tres capas.
• Distinción entre *bloques libres* y *protegidos* en el código para impedir la modificación del código generado.

Comparaciones

Variables Utilizadas: 1. Licencia, 2. Página oficial, 3. Compañía, 4. Configuración mínima del equipo, 5. Configuración recomendada del equipo, 6. Sistemas Operativos Soportados, 7. Ediciones, 8. Plataforma, 9. Lenguaje de Modelado, 10. Documentación/Ayuda, 11. Ciclo de vida, 12. Soporte para PIM's, 13. Soporte para PSM's, 14. Permite varias implementaciones, 15. Interoperabilidad, 16. Trazabilidad, 17. Permite la transformación de modelos a otros modelos, 18. Calidad del código generado, 19. Integración de Modelos, 20. Verificador de Modelos, 21. Facilidad en la creación de aplicaciones, 22. Qué elementos UML permite, 23. Transformaciones.

	Andromeda	ArcStyler	OptimalJ
1	Open Source.	Comercial.	Comercial.
2	www.androMDA.org/	www.arcstyler.com	www.compuware.com
3	Gentleware.	Interactive Objects Software.	Compuware.
4	• CPU 1,2 GHz. • 512 MB de Memoria. • 400 MB de espacio libre en disco.	• CPU 1,4 GHz. • 512 MB de Memoria. • 400 MB de espacio libre en disco (800 MB durante la instalación), dependiendo del paquete de MDACartridges que se instale.	• CPU 1 GHz. • 512 MB de Memoria. • 450 MB de espacio libre en disco. • *Software* de requisito previo requerido: JDK 1.4.2 o 5.0. • Resolución de Pantalla de 1280 x 1024 pixel.
5	• CPU 2 GHz. • 1 GB de Memoria o más. • 1 GB de espacio libre en disco. • Resolución de Pantalla de 1280 x 1024 pixel.	• CPU 2 GHz. • 1 GB de Memoria o más. • 1 GB de espacio libre en disco. • Resolución de Pantalla de 1280 x 1024 pixel.	Los requisitos mínimos permiten utilizar el producto para desarrollar y para correr pequeñas aplicaciones (incluyendo web server, motor del servlet y servidor de EJB). Para usos más grandes se recomienda tener un computador con una CPU más rápida y con más memoria.
6	• Windows 2000. • Windows XP. • Red Hat 8.0. • Windows 2003 Server.	• Windows NT 4.0. • Windows 2000. • Windows XP. • Windows 2003 Server. • Linux (Probado en SuSE 9.2 y GenToo),	• Windows 2000. • Windows XP. • Red Hat 8.0.
7	AndroMDA.	ArcStyler Enterprise Edition o ArcStyler Architect Edition.	OptimalJ Architecture Edition, OptimalJ Professional Edition y OptimalJ Developer.
8	J2EE, Spring, .NET	J2EE y .NET	J2EE
9	UML (MagicDraw, Poseidon, ArgoUML entre otras).	UML (MagicDraw).	UML (MagicDraw).

	Andromeda	ArcStyler	OptimalJ
10	Dispone de numerosos manuales para el manejo del aplicativo (en su gran mayoría en inglés).	Dispone de numerosos manuales tanto para el manejo del aplicativo como para el uso de MDA Cartridge (en su gran mayoría en idiomas diferentes al español).	Dispone de numerosos manuales para el manejo del aplicativo (en su gran mayoría en idiomas diferentes al español).
11	Soporta casi todo el ciclo de vida (excepto Soporte para el manejo de requisitos).	Soporta casi todo el ciclo de vida (excepto Soporte para codificación y despliegue).	Soporta casi todo el ciclo de vida (excepto Soporte para el manejo de requisitos).
12	Posee soporte para especificar sistemas mediante PIM's y se hace a través del Modelo de Clases.	SI (Mediante Diagramas de Clases UML).	Posee un fuerte soporte para especificar sistemas mediante PIM's y está representado por el Modelo de Clases.
13	No tiene soporte para construir PSM's explícitos, se usa un PIM, los cartuchos y los perfiles UML para generar directamente el código.	NO (Ante la falta de un PSM explícito, utiliza un PIM estereotipado y las marcas para generar el código directamente).	Genera a partir del PIM tres tipos de PSM (Web, Bases de Datos y EJB).
14	Incorpora implementaciones para generar código Java (en particular J2EE), y gracias a su sistema de cartuchos puede ampliarse a cualquier otra plataforma.	SI (Java2, EJB, Servicios Web, Corba, .NET y gracias a CARAT permite definir nuevos MDA-Cartridge).	Genera tres tipos distintos de PSM's a partir del mismo PIM, pero todas van dirigidos a la misma plataforma J2EE.
15	SI (Importa y Exporta modelos a través de XMI).	SI (Importa y Exporta modelos a través de XMI).	SI (Importa y Exporta modelos a través de XMI).
16	La herramienta ofrece poco soporte en este aspecto.	No cuenta con un registro que permita conocer qué elemento del PIM corresponde al del PSM.	Permite conocer qué elemento del PIM le corresponde a un elemento del PSM y el sitio donde está ubicado en el código.
17	Permite la transformación de modelo a modelo ayudando así a levantar el nivel de abstracción, y adicionalmente permite escribir las propias transformaciones del usuario.	Posee un módulo que permite crear y extender las transformaciones según las necesidades específicas.	Esta herramienta proporciona los mecanismos de transformación entre modelos.

	Andromeda	ArcStyler	OptimalJ
18	Bien documentado y legible.	Bien documentado y legible.	Bien documentado pero poco legible ya que utiliza varios patrones en el código.
19	Para las transformaciones de unos modelos a otros cada uno de ellos es contenido en un repositorio de metadatos.	Permite que los elementos de los modelos interactúen entre sí, pero con algo de dificultad debido a que un cartucho no siempre es capaz de crear puentes para comunicarse con los elementos creados por otro cartucho.	Los PSM's (Web, Bases de Datos y EJB) se integran perfectamente entre sí de forma transparente y automática.
20	Valida los modelos de entrada usando OCL que se relacionan con las clases del metamodelo.	Cada cartucho debe encargarse de implementar el verificador de sus modelos de entrada. Muchos errores pasan desapercibidos para el verificador y se trasladan a la fase de generación de código.	Incluye verificadores de modelos para PIM y para PSM's que se ejecutan antes de una transformación, advirtiendo al programador de posibles errores en la generación de código.
21	Se requiere de un buen conocimiento de la herramienta para poder desarrollar.	Requiere de un mayor esfuerzo de desarrollo, el programador está en la obligación de escribir el código de integración de modelos.	En poco tiempo permite crear una aplicación a partir de un simple modelo de clases, generando código de calidad y aplicando patrones de diseño.
22	• Diagramas de clases. • Restricciones. • OCL. • Casos de uso. • Diagramas de actividad. • Diagramas de estado. • Diagramas de secuencia. • Diagramas de colaboración. • Diagramas de componentes. • Diagramas de despliegue.	• Diagramas de clases. • Restricciones. • OCL. • Casos de uso. • Diagramas de actividad. • Diagramas de estado. • Diagramas de secuencia. • Diagramas de colaboración.	• Diagramas de clases. • Restricciones. • OCL. • Casos de uso. • Diagramas de actividad. • Diagramas de estado. • Diagramas de secuencia. • Diagramas de colaboración. • Diagramas de componentes. • Diagramas de despliegue.
23	La aplicación permite tanto la creación de nuevos cartuchos (permitiendo usar hasta la propia herramienta para generarlos ya que son archivos .JAR) así como también la extensión de los existentes.	Permite tanto la creación de nuevos MDA-Cartridge por medio de CARAT así como también la extensión de los existentes.	La edición OptimalJ Professional no permite la modificación de las transformaciones pero la edición OptimalJ Architecture brinda control absoluto sobre ellas.

Conclusiones

Con el surgimiento de *MDA* se ha dado un nuevo impulso a la generación automática del código de una aplicación a partir de su especificación. *MDA* automatiza las tareas de diseño, desarrollo y en buena parte despliegue de las aplicaciones. Las herramientas *MDA* que se eligieron para este comparativo son Andro*MDA*, ArcStyler y OptimalJ; a continuación se resumen las conclusiones obtenidas a partir del comparativo. Esto podría llevar en un futuro a anular la etapa de codificación, permitiendo al ingeniero centrarse en un desarrollo de alto nivel.

AndroMDA: tiene una ventaja sobre las otras dos herramientas comparadas y es que es *software* Open Source, lo que permite que programadores a nivel mundial hagan contribuciones permanentes al mejoramiento de esta herramienta; cuenta con un número apreciable de *plug-ins* y cartuchos que generan código a un gran número de plataformas. Aunque se basa en *MDA*, no basta sólo con los modelos para llegar a un despliegue, es necesario que el desarrollador intervenga el código y por lo tanto requiere que éste tenga un buen conocimiento de la plataforma.

OptimalJ: gracias a la clara separación entre el *PIM* y los *PSM* OptimalJ puede considerarse como una herramienta que refleja fielmente el proceso *MDA*, con una clara separación entre el *PIM* y los *PSM*. A partir de un modelo de clases permite crear de forma sencilla y en poco tiempo, una aplicación básica para la plataforma J2EE, generando código de buena calidad.

ArcStyler: en ArcStyler el *PIM* pasa directamente a código sin que exista un *PSM* que pueda cambiar el desarrollador, sin embargo tiene un punto a favor, al igual que Andro*MDA* es una herramienta que permite generar código para diferentes plataformas mediante la arquitectura CARAT (Cartuchos *MDA*). En ArcStyler el esfuerzo de desarrollo es mayor, ya que el programador utiliza *PIM*'s "marcados" en reemplazo de *PSM*'s así como también la creación de etapas intermedias para poder llegar al modelo de destino (código o *PSM*).

Bibliografía

Arlow Jim, Neustadt Ila. (2004). *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML.* Addison-Wesley.

Corredera de Colsa Luis Enrique. "Arquitectura dirigida por modelos para J2ME, 2007". Documento electrónico: http://personal.telefonica.terra.es/web/lencorredera/mda_j2me.pdf [Consultado: Marzo 20 de 2009].

Frankel David S. (2003). *Model Driven Architecture Applying MDA to Enterprise Computing.* Wiley Publishing.

Kleppe Anneke, Warmer Jos, Bast Wim. (2005). *MDA Explained: The Model Driven Architecture: Practice and Promise.* Addison-Wesley.

Mellor Stephen J, Scott Stephen J, Uhl Axel, Weise Dirk. (2004). *MDA Distilled: Principles of Model-Driven Architecture.* Addison-Wesley.

Warmer Jos, Kleppe Anneke. (2003). *Object Constraint Language, The: Getting Your Models Ready for MDA.* Second Edition. Pearson Education.