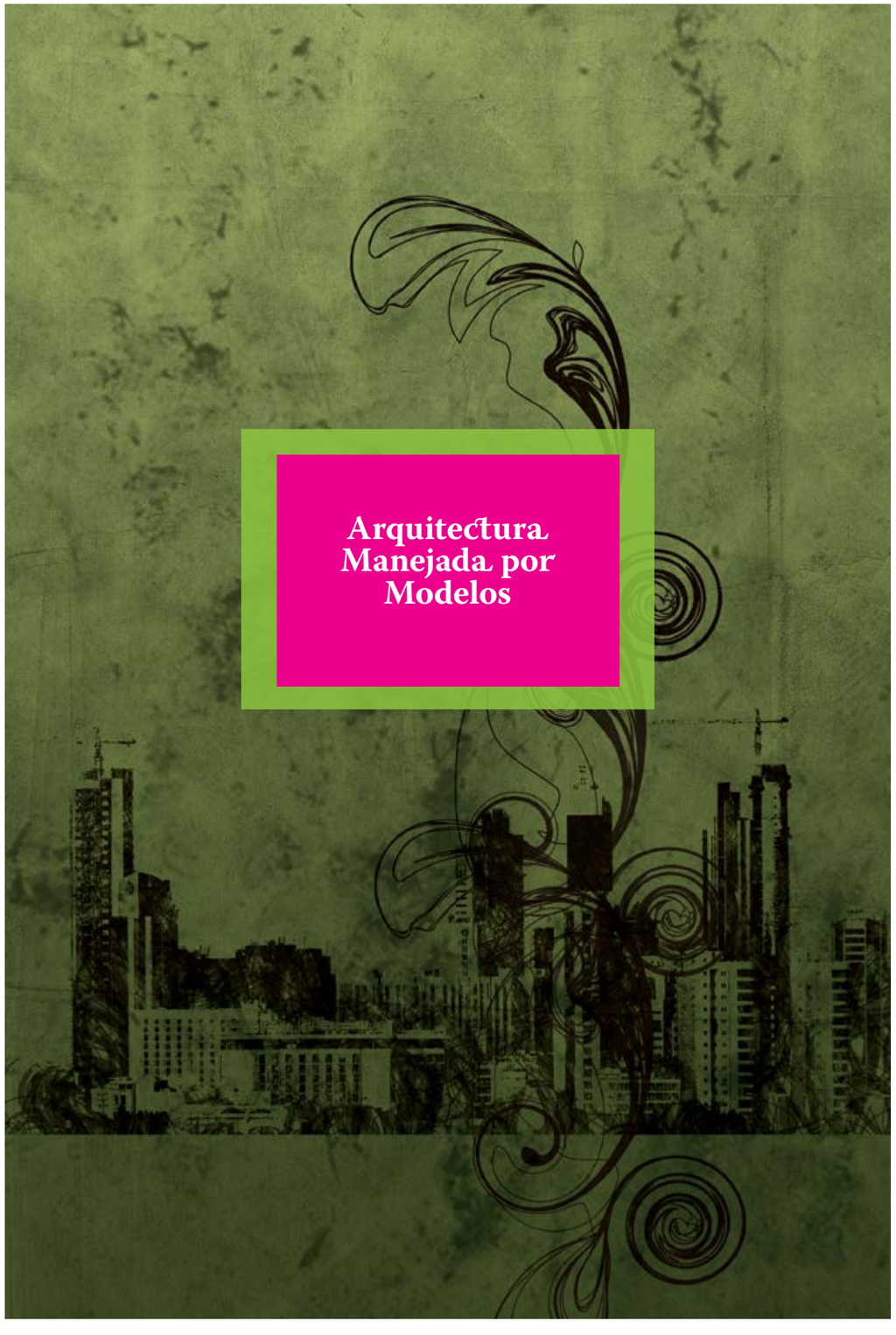

Arquitectura Manejada por Modelos

RESUMEN

En el desarrollo del software es fundamental potenciar la reutilización de sus elementos con el fin de facilitar la labor de los diversos roles que participan del proceso. La arquitectura dirigida por modelos (MDA) describe o propone un proceso de desarrollo basado en la transformación de modelos. Los principios en que se fundamenta MDA son la abstracción, la automatización y estandarización. "El proceso central de MDA es la transformación de modelos que parten del espacio del problema (CIM) hasta modelos específicos de plataforma (PSM), pasando por los modelos que describen una solución independiente de la computación (PIM)".

Para entender el papel de los modelos en el proceso de desarrollo de software este trabajo describe el funcionamiento, los estándares y los métodos de transformación que propone MDA.

PALABRAS CLAVES: MDA; Software; Modelos.

ABSTRACT

In software development is essential to enhance the reuse of software components in order to facilitate the work of the various roles involved in the process. Model Driven Architecture (MDA) describes or proposes a development process based on model transformation. The principles which underpin MDA are the abstraction, automation and standardization. "The central process of MDA is the transformation of models that starts from the Computer Independent Model (CIM) to Platform Specific Models (PSM), to models that describes a Platform Independent Model (PIM)."

To understand the role of models in the software development process, this paper describes the operation, standards and processing methods proposed by MDA.

KEYWORDS: MDA; Software; Models.

Leopoldo Vinicio Venegas Loor

Universidad Particular San Gregorio de Portoviejo

lvvenegas@sangregorio.edu.ec

ARTÍCULO PRESENTADO PARA REVISIÓN: 21 DE AGOSTO DE 2014
ARTÍCULO ACEPTADO PARA PUBLICACIÓN: 23 DE SEPTIEMBRE DE 2014

INTRODUCCIÓN

Las Infraestructuras informáticas están ampliando su alcance en todas sus dimensiones. Las nuevas plataformas y las aplicaciones deben interoperar con los sistemas heredados. Las empresas virtuales abarcan varias compañías; el Internet está imponiendo nuevos retos de integración, ya que se extiende a todos los rincones de cada organización. Nuevas plataformas de implementación están continuamente desarrollándose, cada cual afirmando ser "la próxima gran solución".

Quienes diseñan sistemas informáticos, ya sea por los bancos u organizaciones de cualquier actividad, se enfrentan a enormes opciones de tecnología. Para proteger sus inversiones y maximizar la flexibilidad, la compra de hardware que implementa los estándares abiertos de interconexión, como Ethernet y USB, y software que utiliza estándares abiertos como el de la interfaz CORBA, que constituye lo único sensato en el rápido mundo cambiante de hoy, y en el medio ambiente de la computación de múltiples proveedores.

Cuando los ordenadores y las redes son más rápidos y más baratos, incluso las normas de interconexión deben evolucionar. Las nuevas tecnologías aparecen constantemente como nichos de nuevas aplicaciones.

Model Driven Architecture (MDA) soporta los estándares de la evolución en los dominios de aplicación de diversas organizaciones, como la planificación de recursos empresariales, control de tráfico aéreo y la investigación del genoma humano. Sin embargo deben sobrevivir a los cambios en la tecnología y la proliferación de diferentes tipos de middleware. El MDA se ocupa del ciclo de vida completo de diseño, despliegue, integración y gestión de aplicaciones, así como los datos mediante estándares abiertos. MDA basados en estándares permite a las organizaciones integrar lo que ya tiene en su lugar con todo lo que se construye hoy y cualquier aplicación que se requiera construir en el mañana.

Al plantear la interrogante sobre cuál es el gran reto de las industrias que desarrollan software nos encontramos con una diversidad de respuestas que parecen converger a la necesidad de mejorar el desempeño para maximizar las ganancias. El Human Performance Center (HPC) agrupa las tendencias que sigue la comunidad informática para lograr tal propósito, en tres enfoques básicos (HPC, 2002):

- Trabajando más rápidamente: mejorando las herramientas que apoyan el desarrollo de software (IDE - Integrated Development Environment), compiladores, generadores de código, etc.

- Trabajando más ágilmente: analizando, evaluando y mejorando la forma de trabajar (SPI-Software Product Line).

- Trabajando menos: cambiando la forma de trabajar, maximizando la reutilización, no desgastándose en dise¬ño, codificación y pruebas exhaustivas, realizando programación en el nivel de ingeniería de modelos y requisitos.

El planteamiento anterior evidencia la im¬portancia de proponer estrategias de trabajo que potencien la reutilización a un alto nivel de abstracción. El reto que en la actualidad motiva a la comu¬nidad de investigadores y generadores de tecnología es proponer esquemas de desarrollo en los cuales los modelos, antes que el código, son los actores cen¬trales del proceso de desarrollo y donde se proveen mecanismos y herramientas de trabajo integradas que asisten al desarrollador en la construcción y transformación progresivas de modelos hasta llegar a la solución final. Esta corriente de trabajo, liderada por el OMG (Object Management Group), se conoce como arquitectura dirigida por modelos (MDA).

El objetivo de este artículo es explorar los conceptos, principios e importancia; entender su arquitectura y el proceso de desarrollo, basado en MDA para citar los métodos, estándares y herramientas que podrían apoyar su apropiación.

I.- LA ARQUITECTURA DIRIGIDA POR MODELOS

La Arquitectura basada por modelos lleva a resolver problemas de tiempo, costos y calidad asociados al desarrollo del software. En este contexto MDA proporciona un marco de trabajo en el cual es posible especificar modelos, en diferentes niveles de abstracción, y pasar desde un modelo a otro por medio de transformaciones. Dichas transformaciones de modelos deben ser expresadas de manera clara y precisa, usando un lenguaje definido para ese propósito.

MDA es una estandarización de la OMG como plataforma para soportar una arquitectura manejada por modelos, Figura 1.

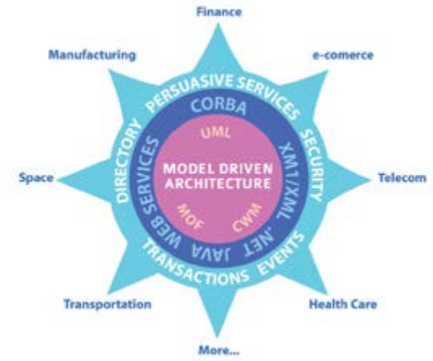

Figura 1. Modelo MDA
Autor: OMG (Object Management Group)

1.1.CONCEPTOS BÁSICOS DE MDA.

Algunos de los conceptos más importantes que forman parte de la especificación mda son:

- MODELO. Es una descripción o especificación mediante un lenguaje visual de un sistema.

- METAMODELO. Es la descripción y especificación de los elementos y reglas que se utilizan para crear modelos semánticamente correctos para un dominio en particular. También puede definirse como un lenguaje de modelado.

- DIRIGIDO POR MODELOS (Model Driven). Se dice que es dirigido (o guiado) por modelos porque provee mecanismos para dirigir el curso del diseño, la construcción, la implementación, la operación, el mantenimiento y la modificación de una aplicación. Es decir, el proceso depende de los modelos.

- ARQUITECTURA. La arquitectura de un sistema es la especificación de las partes y conectores del mismo, así como las reglas de interacción de estas.

- VISTA. Es una representación del sistema desde un punto de vista determinado.

- 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.

- PUNTO DE VISTA. Un punto de vista en un sistema es una técnica de abstracción que utiliza un conjunto selecto de conceptos arquitecturales y reglas de estructuración, de manera que se enfoque la atención sólo en un problema particular del sistema. MDA especifica tres tipos de puntos de vista sobre un sistema: el punto de vista independiente de la computación, el punto de vista independiente de la plataforma y el punto de vista específico de la plataforma.

- PUNTO DE VISTA INDEPENDIENTE DE LA COMPUTACIÓN. Este se enfoca en el ambiente del sistema y los requerimientos del mismo, es decir, la lógica del negocio. Los detalles de la estructura y el procesamiento del sistema están escondidos o no han sido determinados.

- PUNTO DE VISTA INDEPENDIENTE DE LA PLATAFORMA. Se enfoca en la operación del sistema mientras oculta los detalles específicos para cierta plataforma, es decir, muestra la parte de la implementación que es idéntica de una plataforma a otra.

- PUNTO DE VISTA ESPECÍFICO DE UNA PLATAFORMA. COMBINA EL PUNTO DE VISTA INDEPENDIENte de la plataforma con el detalle del uso de una plataforma especifica.

- MODELO INDEPENDIENTE DE LA COMPUTACIÓN (CIM Computer Independent Model). Es una descripción de la lógica del negocio desde una perspectiva independiente de la computación. Es un modelo del dominio.

- MODELO INDEPENDIENTE DE LA PLATAFORMA (PIM Platform Independent Model). Es una descripción de la funcionalidad del sistema en forma independiente de las características de plataformas de implementación específicas.

- MODELO ESPECIFICO A UNA PLATAFORMA (PSM Platform Specific Model). Es una vista del sistema desde que hace referencia a una plataforma específica. Un PSM combina las especificaciones en el PIM con los detalles de cómo el sistema utiliza un tipo de plataforma en particular. Por ejemplo, .NET, J2EE, relacional.

- MODELO ESPECÍFICO DE IMPLEMENTACIÓN (ISM (Implementation Specific Model) Es una descripción (especificación) del sistema a nivel de código. Por ejemplo, Java, C#, etc.

- TRANSFORMACIÓN DE MODELOS. Es el proceso de convertir un modelo a otro del mismo sistema. Generalmente el PIM es combinado con alguna información adicional para producir un PSM.

1.2. LOS PRINCIPIOS DE MDA

Cuatro principios subyacen en la opinión de la OMG de la MDA:

1. Los modelos se expresan en una notación bien definida, son una piedra angular para entender los sistemas de soluciones de escala empresarial.

2. La construcción de los sistemas se pueden organizar en torno a un conjunto de modelos mediante la imposición de una serie de transformaciones entre modelos, organizados

REVISTA SAN GREGORIO, 2014, No.8, VOLUMEN 2, JUNIO-DICIEMBRE, (64-73), ISSN 1390-7247

en un marco arquitectónico de las capas y las transformaciones.

3. Un respaldo formal para describir los modelos en un conjunto de metamodelos facilita la integración y la transformación significativa entre los modelos, y es la base para la automatización a través de herramientas.

4. La aceptación y la adopción generalizada de este enfoque basado en el modelo requiere de estándares de la industria para proporcionar transparencia a los consumidores, y fomentar la competencia entre los vendedores.

II.- IMPORTANCIA DE LOS MODELOS EN MDA.

El concepto del modelado, como una de las estrategias básicas del desarrollador para entender un problema y proponer una solución, es amplia¬mente aceptado en la ingeniería de software, mucho antes del surgimiento de MDA. Sin embargo, la apli¬cación del modelado en la práctica diaria presenta problemas como los enunciados a continuación:

- Los modelos se usan solo como documentación. Los modelos no funcionan como un artefacto ac¬tivo que contribuya en el proceso de desarrollo.
- Existen vacíos entre el modelo y la implementa¬ción de los sistemas. Los cambios en el modelo no se reflejan en el código y los cambios en el código no se reflejan en los modelos, sólo se genera el código de los modelos la primera vez y nunca se actualiza.
- No hay una adecuada mezcla de modelos. Vis¬tas desconectadas de un sistema (desconexión horizontal) y grupos de modelos desconectados (desconexión vertical).
- No hay transformación de modelos. Pocos lengua¬jes de transformación populares y pocas herra¬mientas que soporten estas transformaciones.

Lo anterior les atribuye a los modelos la fama de una costosa y pesada carga que complica la labor de los participantes en el proceso de desarrollo. La MDA rescata la importancia de los modelos como estrategia clave para entender y especificar una solución de software y progresivamente obtener la solución final.

El uso de una Arquitectura manejada por modelos aporta al producto final las siguientes características: mejorar la productividad, la portabilidad, la interoperabilidad y la reutilización de los sistemas.

III.- EL PROCESO DE DESARROLLO BASADO EN MDA

El desarrollo de software dirigido por modelos (MDD), y más concretamente la propuesta MDA de OMG, constituyen una aproximación para el desarrollo de software, basada en la separación entre la especificación de funcionalidad esencial del sistema y la implementación de dicha funcionalidad usando plataformas de implementación específica.

A grandes rasgos, el proceso de desarrollo de software con MDA se puede dividir en tres fases:

1. Construcción de un Modelo Independiente de la Plataforma (Platform Independent Model o PIM), es un modelo de alto nivel del sistema independiente de cualquier tecnología o plataforma.

2. Transformación del modelo anterior a uno o varios Modelos Específicos de Plataforma (Platform Specific Model o PSM), un PSM es un modelo de más bajo nivel que el PIM que describe el sistema de acuerdo con una tecnología de implementación determinada.

3. Generación de código a partir de cada PSM, debido a que cada PSM está muy ligado a una tecnología concreta, la transformación de cada PSM a código puede automatizarse.

El paso de PIM a PSM y de PSM a código no se realiza manualmente sino empleando herramientas de transformación para automatizar las tareas. La Figura 2 muestra de manera resumida el proceso de desarrollo con MDA, suponiendo que tenemos un único PSM.

Así, MDA es un estándar que promueve a MDD y agrupa a varios lenguajes que pueden ser utilizados para seguir un enfoque dirigido por modelos en una organización, MDA intenta estandarizar MDD, que durante muchos años ha estado a la deriva. MDA no define técnicas, etapas ni artefactos pero si proporciona una estructura tecnológica y conceptual para poder implementar de manera correcta MDD.

3.1. FUNCIONAMIENTO DEL MDA

Hoy en día existe mucha información sobre cómo traducir, por compilación o interpretación, un lenguaje de alto nivel (Java o SQL) a operaciones que un microprocesador es capaz de ejecutar. Así también, en MDA existen “compiladores” capaces de traducir modelos de datos o de la aplicación basados en UML a lenguajes de alto nivel y por lo tanto a las distintas plataformas de los sistemas actuales, pero más importante, a las plataformas del futuro. Como por ejemplo sucede en la industria automotriz, donde la mayor parte del proceso de desarrollo de nuevos vehículos se hace en computadoras y simuladores, para luego construir las partes y ensamblar los vehículos de forma automatizada. En la industria de software sucede algo parecido: el proceso de

desarrollo de software se basa en modelos en una computadora, los cuales, inicialmente, serán PIMs y mediante transformaciones hechas por la computadora, poder generar los PSMs para una o varias plataformas, y al final transformar éstos a código que implemente la solución descrita en los modelos.

Si las tecnologías cambian o se innovan, solo hay que transformar los modelos independientes de la plataforma a los modelos específicos de la nueva plataforma y regenerar la aplicación. Si la aplicación requiere integrarse con otras aplicaciones, se modifican los modelos, y después se regeneran las aplicaciones. Así, entre las metas de MDA se tienen: la portabilidad, la interoperabilidad y la reutilización. La Figura 3 resume el proceso de desarrollo utilizando MDA.

En MDA un desarrollador sólo crea PIMs que son interpretados por una computadora para generar PSMs y posteriormente el código para distintas plataformas.

Figura2. Funcionamiento de MDA.
Elaboración: fuente propia.

De manera más sintética, los pasos a seguir en el proceso MDA de desarrollo de software son los siguientes:

1. Definir un CIM que muestre el sistema dentro del entorno en el que va a operar. Este modelo ayuda a entender exactamente lo que el sistema va a hacer independientemente de cómo se implementa.

2. Construir un PIM, que describe el sistema, pero no muestra los detalles de su implementación en ninguna plataforma.

3. En este estadio del proceso el arquitecto de software ha de elegir una o varias plataformas que permitan la implementación del sistema con las cualidades arquitectónicas deseadas.

4. El arquitecto marca los elementos del PIM para indicar los mappings que han de ser usados para llevar a cabo la transformación de ese PIM en un PSM; o bien, si se utilizan transformaciones de metamodelos, se utilizará una máquina de transformación.

5. Transformar el PIM marcado en un PSM -que puede ser realizado manualmente o con ayuda de una herramienta-: la entrada de la transformación será el PIM marcado y el mapping; la salida es el PSM y el registro de la transformación.

6. Un PSM puede proporcionar más o menos detalle, dependiendo de su propósito, ya que un PIM puede ser una implementación si proporciona toda la información necesaria para construir un sistema y ponerlo en operación, o bien puede ser el PIM de la siguiente iteración del proceso MDA hasta que se consiga una implementación adecuada del sistema.

IV. ESTÁNDARES EN MDA

El papel de los estándares en MDA: los estándares relevantes en los que se apoya la propuesta de MDA tienen el propósito de lograr la interoperabilidad de las herramientas y plataformas. A continuación se detallan los siguientes:

- Arquitectura de cuatro niveles (MOF). Esta arquitectura de modelos denominada MOF (Meta Object Facility) representa el nivel meta más general y tiene el objetivo de permitir la incor¬poración de nuevos lenguajes de modelado (meta-modelos) para fines específicos. Sus componentes o capas de esta arquitectura son: Capa M3 (Metametamodelo), Capa M2 (Metamodelos), Capa M1 (Modelos), Capa M0 (Instancias).

- Lenguaje de restricciones de objetos (OCL). El OCL (Object Constraint Language) es un lenguaje para especificar restric¬ciones sobre los objetos de un modelo UML (Cernu¬da, 2002; OMG RfPOCL, 2003). Uno de los principales objetivos que se persiguen con OCL es precisar la especificación de los modelos escritos en UML, evitando ambigüedades en su interpretación y garantizando su consistencia semántica.

- Perfiles. Los perfiles han tomado relevancia e importancia por su gran utilidad en los procesos de transformación de modelos. Un perfil está constituido por estereotipos, valores etiquetados y restricciones, sirve para extender la semántica de los modelos construidos con UML a un dominio específico o a una plataforma particular.

- XMI (XML Metadata Interchange). El XMI es un lenguaje para permitirles a los usuarios desarrolladores de software el intercambio de modelos UML. Es una especificación del OMG definida con el objetivo de facilitar el intercambio de modelos entre herramientas de modelado.

V. MÉTODOS DE TRANSFORMACIÓN EN MDA.

Métodos específicos de transformación de modelos:

- Transformación por marcado. En un modelo son colocadas una serie de marcas que serán utilizadas para guiar la transformación (Figura 3). Una vez que se tiene el modelo marcado, se aplican reglas de mapeo definidas para una plataforma específica para transformar el PIM en un PSM.

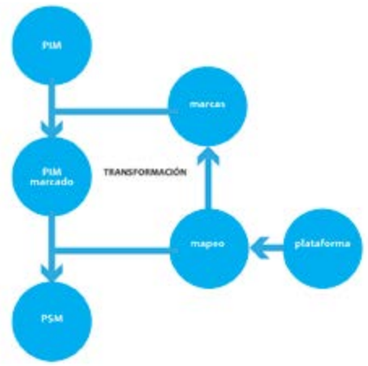

Figura 3. Transformación por Mercado.
Autor: Claudia Pons - Facultad de Informática. Universidad Nacional de la Plata 2010.

- Transformación por metamodelo. Un modelo es construido usando un lenguaje independiente de la plataforma especificado por algún metamodelo. Se hace la transformación de este modelo a un lenguaje específico de la plataforma especificado en algún otro metamodelo. Es decir se transforma de un lenguaje de modelado a otro (Figura 4). Se tiene un lenguaje de modelado fuente que será transformado en un lenguaje de modelado objetivo.

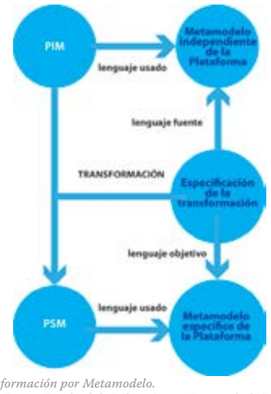

Figura 4. Transformación por Metamodelo.
Autor: Claudia Pons - Facultad de Informática. Universidad Nacional de la Plata 2010.

- Transformación por patrones. Patrones pueden ser utilizados en la especificación del mapeado (Figura 5). El mapeado incluirá entonces patrones y marcas correspondientes a elementos de dichos patrones.

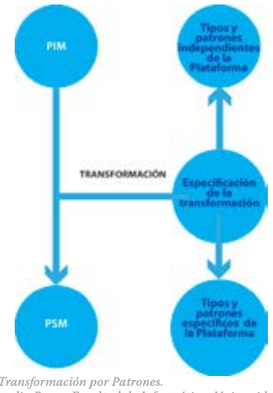

Figura 5. Transformación por Patrones.
Autor: Claudia Pons - Facultad de Informática. Universidad Nacional de la Plata 2010.

Se debe notar que sea cual sea el método de transformación seleccionado para pasar de un modelo a otro se debe determinar un conjunto de reglas de transformación, en las cuales se indica que elementos del modelo nuevo serán generados tomando como base elementos del modelo original. Así el mapeo consiste en determinar estas reglas de transformación, mapear los elementos de un modelo a los de otro, y de esta forma, facilitar el proceso de transformación.

VI. HERRAMIENTAS PARA MDA.

El uso de herramientas MDA permite principalmente:

- Reducir el costo de implementación.
- Mitigar los riesgos.
- Incrementar la agilidad del negocio al reducir el tiempo de implementación.
- Proteger la inversión de la tecnología debido a la oportunidad de automatización.

En MDA las herramientas de transformación son esenciales sin embargo no son las únicas, a continuación se citan e ilustran los diferentes tipos de herramientas (figura 7):

- Editor de código (IDE). Proveen funciones de depuración, compilación y edición de código.
- Repositorio de modelos.
- Editor de modelos.- Herramienta CASE donde los modelos pueden ser construidos y modificados.
- Validador de modelos. Los validadores revisan los modelos contra un conjunto de reglas (predefinidas o definidas por el usuario) para asegurar que el modelo está listo y pueda ser usado en una transformación.

- Editor de definición de transformaciones para crear y modificar una definición.
- Repositorio de definiciones de transformación que son utilizadas para pasar de un modelo a otro.
- Ejecución de modelos. Motor que provee una plataforma virtual donde se pueden ejecutar los modelos.

Figura 6. Herramientas MDA.
Autor: Claudia Pons - Facultad de Informática. Universidad Nacional de la Plata 2010.

En el mercado se destacan las siguientes herramientas tanto libres como comerciales:

- AndroMDA.
- OptimalJ.
- ArcStyler

VII. IMPACTO DE MDA EN ESTRUCTURA DE IT.

La apropiación de MDA en la Ingeniería de Software genera productividad a sus protagonistas, conduciendo a no solo obtener un código que funciona sino brindando eficiencia al proceso. A continuación las nuevas características de cada rol:

- Analista de Requerimientos.- Los requerimientos se levantan empleando modelos UML que restan ambigüedad y facilitan la visualización, aun cuando su creación no pueda ser automatizada.
- Arquitectos.- Adicional a elegir la estructura del sistema sus actividades cambian a perfeccionar y reutilizar los modelos.
- Analista Programador.- Su trabajo trasciende de escribir código a crear y afinar patrones de código.
- Testeador.- El uso de herramientas de pruebas lo hace más productivo alcanzando un alto número de pruebas individuales, en contraposición a la escritura de scripts de prueba y sentencias de código.
- Mantenedor.- Usualmente el mantenimiento representa la mitad del TCO del producto de software de larga vida, MDA extiende la vida del sistema reduciendo la carga de actividades de mantenimiento y focalizando su acción al mantenimiento del modelo y diseño de reglas del negocio, más no al del código
- Clientes.- El producto final para los usuarios es mejor.

CONCLUSIONES

En base a la Investigación realizada, se pudo conocer que todas las formas de Ingeniería se basan en el uso de modelos para facilitar la comprensión y desarrollo de sistemas complejos. En este contexto, la Arquitectura manejada por Modelos (MDA), se fundamenta en tres aspectos de gran importancia la abstracción, la estandarización y la automatización de procesos dentro del desarrollo en la Ingeniería del Software, que permite obtener beneficios como portabilidad, interoperabilidad y reusabilidad, restando además complejidad al desarrollo del software. ◆

REFERENCIAS BIBLIOGRÁFICAS

◆ *Ambler Agile, S.W. (2004) Model Driven Development with UML. Cambridge: University Press.*

◆ *De Lara, Juan. (ff.aa.) Model Driven Architecture.*

◆ *Disponible: http://astreo.ii.uam.es/~jlara/doctorado.2006/tema5.pdf*

◆ *Disponible: http://danielguzman.wordpress.com/2010/09/13/modelo-comparativo-de-herramientas-mda-en-ambientes-de-software-libre/*

◆ *Garcia, J., Rodriguez, J, Menarquez, M; Ortin, M., Sanchez, J. (ff.aa) Un estudio comparativo de dos herramientas MDA: optimalj y arcstyler.*

◆ *Disponible: http://dis.um.es/~jmolina/tdsdm04jesus_final.doc*

◆ *Guzmán, Daniel (ff.aa.) Modelo Comparativo de Herramientas MDA en Ambientes de Software Libre.*

◆ *Mellor, Stephen; Watson, Andrew. Roles in the MDA Process*

◆ *Disponible: http://www.omg.org/registration/Roles_in_MDA1.pdf*

◆ *Pelayo García-Bustelo, B. C. y Cueva Lovelle, J. M.. "Arquitecturas dirigidas por modelos (MDA). El framework C3NET". II Congreso Internacional de Ingeniería de Computación y Sistemas (IICIIS). Trujillo, PERÚ, 2006.*